PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30 — A1

(11) International Publication Number: WO 97/15890

(43) International Publication Date: 1 May 1997 (01.05.97)

(21) International Application Number: PCT/US96/17142

(22) International Filing Date: 25 October 1996 (25.10.96)

(30) Priority Data:
08/549,359 27 October 1995 (27.10.95) US

(71) Applicant: AT & T CORP. [US/US]; 32 Avenue of the Americas, New York, NY 10013-2412 (US).

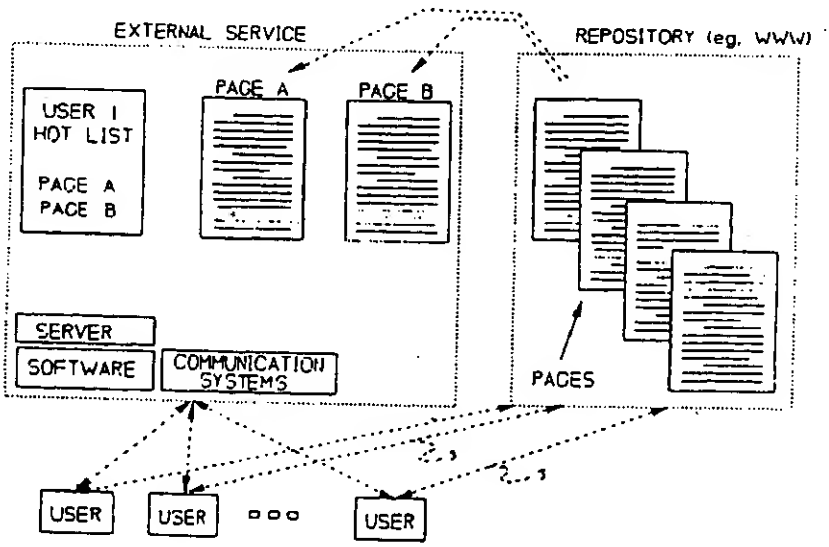
(72) Inventors: BALL, Thomas, J.; 1236 Whitingham Circle, Naperville, IL 60540 (US). DOUGLIS, Frederick; 248 Harlech Way, Somerset, NJ 08873 (US).

(74) Agent: DWORETSKY, Samuel, H.; c/o H.T. Brendzel, AT & T Corp., P.O. Box 4110, Middletown, NJ 07748 (US).

(81) Designated States: CA, JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: IDENTIFYING CHANGES IN ON-LINE DATA REPOSITORIES



(57) Abstract

A system for accessing documents contained in a remote repository, which change in content from version-to-version. The system allows users to specify lists of documents of interest. Based on the lists, the system maintains an archive, which contains a copy of one version of each listed document, and material from which the other versions can be reconstructed. The system periodically compares the archive with current versions of the documents located in the respository, and updates the archive, thereby maintaining the ability to reconstruct current versions. The system also monitors access to the versions by each user. When a user calls for a current version, the system presents the current version, and indicates what parts of the current version have not been previously accessed by the user.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia
AT	Austria
AU	Australia
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CS	Czechoslovakia
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgystan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakhstan
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam

IDENTIFYING CHANGES IN ON-LINE DATA REPOSITORIES

The invention concerns presentation of a current version of a document retrieved from a data repository. The presentation indicates changes made in the document since the viewer accessed a previous version.

BACKGROUND OF THE INVENTION

Information which is stored in computerized systems can change frequently, and without notice. As an example, software under development frequently involves many persons, and is commonly stored at a central location. Each person can change the software on an ad hoc basis, without knowledge of others.

In such systems containing changeable data, a person who examines information on a given day does not, in general, know whether, and how, the information has changed since a previous examination. Consequently, the person must spend time comparing currently available information with previous versions of the information.

Software exists for facilitating this comparison. For example, systems known as "version control systems," or "revision control systems," store data which represents multiple versions of different documents, as indicated in Figure 1A. In that Figure, the DATA is indicated, together with dashed loops which indicate the VERSIONs.

The loops indicate that the VERSIONs are contained in, and

MISSING UPON TIME OF PUBLICATION

MISSING UPON TIME OF PUBLICATION

interspersed among a much larger document. Small arrows point to changes, which are primarily additions in this case. The change in the "last update" date give an example of text being replaced. Here the page's author had highlighted the changes manually with small icons as well. The banner at the top of the page was inserted by HTMLDIFF.

Figure 12 illustrates version histories which give the user a chance to compare any two versions, or to go directly to a selected version.

Figure 13 illustrates output of W3NEWER, and shows a number of anchors (the descriptive text originates from the hot list). The anchors marked "changed" have modification dates after the time which the user's browser history indicates the URL was last seen. Some URLs were not checked at all, and others were checked and are known to have been seen by the user.

Figure 14 demonstrates use of a SNAPSHOT facility, which allows a user to specify an operation on a URL. In this example, DOUGLIS@RESEARCH.ATT.COM is "remembering" URL HTTP://SNAPPLE.CS.WASHINGTON.EDU:600/MOBILE/.

DETAILED DESCRIPTION OF THE INVENTION

A TECHNICAL APPENDIX, which is located at the end of this Specification, describes the invention in detail. Following the TECHNICAL APPENDIX is a COMPUTER PROGRAM LISTING, which contains code which implements one form of the invention.

An illustrative embodiment of the invention is given in the

discussion below.

Overview of Invention

A commonly used repository of information is known as the World Wide Web, or WWW. In the WWW, providers of information make their information available to users in the form of "pages." Each page is assigned a name, which distinguishes the page from other pages, and allows a user to locate the page.

The WWW provides information using an information retrieval-and-display approach called "hypertext." In hypertext, a page may contain references to other pages, or other documents. A user can call up a page which is referenced, by clicking on the reference (called a URL, or Universal Resource Locator) with a pointing device. Figure 1B provides an example.

In Figure 1B, a document D is displayed to a user. References R refer to other documents. For example, R1 refers to D1, R2 refers to D2, and so on. The referenced documents themselves may contain their own references to other documents, such as R4, which refers to D4.

A user can retrieve a referenced document D, by clicking on the reference R which refers to it. For example, clicking on R1 causes retrieval and display of D1.

Under the invention, a user of the WWW initially identifies pages of interest. Document D in Figure 1B represents one page. These selected pages form a "hot list." Then, the invention does the following:

(a) Copies the hot-listed pages into an archive, which is a storage location separate from the WWW, and under independent control. After the copying, the original pages continue to reside in the WWW, and copies reside in the archive.

(b) Monitors, at later times, the original pages for changes, and archives the changes.

(c) Records the times when the user later accesses each hot-listed page.

(d) Whenever the user accesses a hot-listed page, presents the user with

i) the current version of the page (which may differ from the initial copy which was stored in the archive); and

ii) an option to compare selected versions of the page. The comparison is presented by performing a differencing operation on pairs of versions.

e) As an option, the invention also implements the steps described above with respect to documents referenced by the page. For example, in Figure 1A, if a user is viewing document D, the invention can present the current version of reference document D2, together with a history of D2.

More Detailed Description

Hot-List Pages are Stored in EXTERNAL SERVICE

Figure 1 illustrates a REPOSITORY of information, such as the WWW. For assistance in accessing the REPOSITORY, the invention provides the EXTERNAL SERVICE which includes:

(a) SOFTWARE, such as that provided in the COMPUTER PROGRAM LISTING herein,

(b) a SERVER, or other computer, which runs the software, and

(c) COMMUNICATION SYSTEMS which link with both the users and the REPOSITORY.

The SERVER and the COMMUNICATION SYSTEMS located within the EXTERNAL SERVICE are known in the art. As indicated in the Figure, the EXTERNAL SERVICE is distinct from the REPOSITORY, and under separate control.

The invention does not disrupt the users' normal interaction with the REPOSITORY; the users can interact with both the REPOSITORY, as usual, and also with the EXTERNAL SERVICE. Dashed arrows 3 indicate the interaction. Several examples will provide illustrative modes of operation of the invention.

Example: Single User

Operation with respect to a single user will first be explained. Figure 2 shows a hot list 4, submitted by USER 1, which identifies pages A and B as being of interest to USER 1. The

invention allows the user to modify the hot list at later times. In response to the hot list, the invention copies pages A and B from the REPOSITORY, as indicated by the dashed arrows. These PAGEs will be termed "base pages." At this time, the originals of PAGEs A and B remain in the REPOSITORY, and copies reside in the EXTERNAL SERVICE.

Then, the invention periodically examines the originals of PAGES A and B, located in the REPOSITORY, for changes. In looking for changes, the invention first performs a preliminary check, based on information such as (1) dates of modification and (2) checksums.

Dates of modification may be added to a PAGE by the PAGE provider. These dates directly indicate whether the originally archived version has changed.

Checksums are generated by the invention. An example of a checksum is the numerical sum of all characters in a line, or on a page. If a checksum changes (indicating that the number of characters has changed), the change indicates a high probability that a change has occurred in the PAGE. (In practice, the checksums used are more complex than this simple example illustrates. Checksums are known in the art.)

If the preliminary check, either by dates of modification or checksums, indicates that changes have occurred, then the invention copies the present version of the PAGE into the EXTERNAL SERVICE, and compares it with the base page, in order to locate the changes. Computer programs for detecting such changes are known in the art,

and some examples are given in the TECHNICAL APPENDIX. A preferred program, not known in the prior art, is entitled W3NEWER, and was developed by the inventors. W3NEWER is contained in the listing located at the end of this Specification.

When changes are found, the invention stores them in the EXTERNAL SERVICE. Figure 3 illustrates storage of the changes, by the small boxes 6 located below PAGEs A and B. The DATEs within the boxes 6 indicate the dates on which the changes were saved.

Figure 3A illustrates how the invention displays the history of versions. Column 7 indicates the number assigned to each version by the invention. Column 8 indicates the times when the respective versions were retrieved by the invention. Column 8A allows a user to select a pair of versions for a differencing operation, as discussed below.

For ease of explanation, Figure 3 illustrates storage of base pages, which are early versions of PAGEs, together with subsequent changes, indicated by the boxes 6. However, in practice, it can be more efficient to perform storage in a reversed sense, by storing the latest version as the base page (instead of the early version) and storing the changes 6 from which early versions can be reconstructed. One reason is that users are expected to call for latest versions more frequently than early versions. Storage of the entire latest versions eliminates the need to reconstruct them.

The changes, together with their base pages, form an archive, which allows reconstruction of a PAGE as of any date desired. For

example:

-- PAGE A itself (ie, the base page), plus the changes labeled DATE 1, allow reconstruction of the version of PAGE A, as of DATE 1.

-- PAGE A itself, plus the changes labeled DATE 1 and DATE 2, allow reconstruction, as of DATE 2, and so on.

When USER 1 wishes to view PAGE A, the invention ordinarily retrieves and presents the current version. The invention also provides an option for reconstructing the PAGE, as of a date specified by the user, and presents it in the format shown in Figure 4. The program HTMLDIFF, contained in the listing, generates the image shown in Figure 4. The content of the page can be divided into three classes.

The first class contains material which has not changed. This class of material is displayed in the font, size, color, and background, as customary in documents downloaded from the REPOSITORY.

The second class represents changes, and contains material not present in the base page, but which has been added. Brackets 9 indicate such material. (The brackets 9 are part of Figure 4, and are not necessarily part of the page generated by the invention.) This material is presented in a particular font, particular size, particular color, and particular background. The choice of these parameters can be varied but, in general, they should be chosen to maximize contrast with the first class of material. In addition to the formatting described immediately

above, the added material is further highlighted by arrows 7.

The third class contains material which was deleted from the base page. Deleted material can be handled in at least three ways. One, deleted material can be simply deleted, so that the page presented to the reader contains no reference to the deleted material.

Two, the deleted material can be deleted, but a reference indicating the deletion is added, such as the phrase "Deleted material occurs here." In this case, the user can be given the option of fetching the deleted, non-visible, material.

Three, deleted material can be presented, but indicated as deleted, as by "redline" format, in which a horizontal line, perhaps red in color, is drawn through the deleted material.

Figure 3B illustrates a display, generated by the invention, which indicates which PAGEs on a user's hot list have undergone changes.

Second Example: Multiple Users

In actual practice, multiple users are expected to use the invention. Each of them submits a hot list. In one approach of the invention, the procedure undertaken for a single user (described above) is repeated for multiple users: all PAGEs, on all hot lists, are copied into the EXTERNAL SERVICE. Then, for each hot list, the originals of the PAGES, located within the REPOSITORY, are monitored for changes, and the changes are retrieved into the EXTERNAL SERVICE, as described above.

However, this approach contains inefficiencies. For example, a given PAGE will probably be identified by more than one hot list. Repeatedly copying that PAGE, for each hot list, would entail storage of multiple copies of the same PAGE. Further, repeatedly comparing the multiple copies with their originals in the REPOSITORY represents a waste of computer time: a single comparison would suffice. The invention reduces these inefficiencies by the approach shown in Figure 5.

This Figure represents a modification of Figure 4, to which a hot list for USER 2 has been added. The added hot list specifies PAGES A and C.

To process the new hot list, the invention first checks whether the PAGEs identified on the added hot list are archived within the EXTERNAL SERVICE. Since PAGE A, plus its changes, are already contained within the archive, that PAGE is not copied. But PAGE C, which is not present in the ARCHIVE, is archived, as indicated by the dashed arrow.

At this time, all PAGEs identified on all hot lists are contained within the archive. To emphasize this fact, PAGE A is indicated twice: once for USER 1, and a second time by a dashed page 14, for USER 2, although, as stated above, PAGE A is stored only once.

After archiving all necessary PAGEs, the originals, located within the REPOSITORY, are periodically monitored for changes, as described above. The changes are copied to the archive of the EXTERNAL SERVICE.

Flow Chart

An exemplary flow chart is shown in Figure 6, which refers to a single-user case. In block 20, the EXTERNAL SERVICE accepts hot lists from users. Then, in block 23, the EXTERNAL SERVICE checks whether the PAGEs identified on the hot lists are contained within the archive. If not, the PAGEs are copied from the REPOSITORY, as indicated by block 26.

Then the logic proceeds to block 29, where the originals of the PAGEs, located in the REPOSITORY, are examined for changes. The examination can include the preliminary checks (for checksums and dates of modification) discussed above. When changes are found, the entire PAGE containing them is downloaded to the EXTERNAL SERVICE, and the changes, indicated by blocks 6 in Figure 3, are derived. Block 32 indicates relevant information stored in the EXTERNAL SERVICE.

As users access the PAGEs, block 35 monitors the times of the accesses, in order to identify which versions of each PAGE the user viewed last. These times are stored, as indicated by block 32 and dashed arrow 37. These times are used to determine which changes in Figure 4 are to be identified as new material, when a PAGE is called by each user. An example will illustrate.

Figure 7, top, illustrates the time-history of changes made to PAGE A. USER 1 accessed this PAGE at time 2, as indicated. Block 35 in Figure 6 monitors and records this time (at TIME 2 in Figure 7, and not earlier, of course).

If USER 1 again accesses the PAGE at time 5, then the invention presents VERSION 1 to the USER. However, if the user accesses the PAGE at time 11, VERSION 2 had been created since the last access by USER 1. The invention had previously identified the changes, and copied them as indicated in Figure 3. Now, at the access at time 11, the invention presents VERSION 1, plus the changes which make VERSION 2, because block 35 in Figure 6 indicates that the USER has not seen VERSION 2.

Returning to the flow chart of Figure 6, block 39 indicates that, when a USER calls for a PAGE, the invention presents the current version, and indicates the changes made (as in Figure 4) since the USER last accessed that page. In the example immediately above, the invention presents VERSION 2 of PAGE A, as in Figure 7, and indicates the changes made since VERSION 1, because VERSION 1 was the last accessed by USER 1.

The flow chart of Figure 6 should not be read as limiting the invention to a linear, sequential mode of operation. In practice, multiple users can present hot lists simultaneously, and other operations shown in the flow chart can also occur together.

Third Example: Notification of Changes

The invention can notify USERs when changes in their hot-listed PAGEs occur, as indicated by the dashed block 40 in Figure 6. This notification can take the form of a flag which is associated with the BASE PAGE in Figure 8. When the USER logs into the EXTERNAL SERVICE, the invention notifies the USER of the

changes to the respective PAGEs. Figure 3B illustrates one approach to identifying PAGEs which have changed.

Other types of notification are possible. For example, the invention need not wait for a user to access a PAGE. The invention can notify the user when changes have been found, as by sending an electronic mail message to the user.

Fourth Example: Common Hot List

The invention can maintain a predetermined hot-list, for a community of users. This hot list contains a list of PAGEs which are considered to be of general interest to the community. This hot list, and the PAGEs identified on it, are made publicly available, to all users, but on a read-only basis. Users cannot modify the hot list, or the pages.

This predetermined hot list can serve as an instructional tool, to educate users in the operation of the invention, and to demonstrate desirable features.

One Architecture of Data Storage

An illustrative approach to storage of the information identified in block 32 of the flow chart of Figure 6 is illustrated in Figure 8, which is explained with reference to Figure 7.

Figure 7 illustrates hypothetical changes to the three PAGEs identified by the two hot lists of Figure 5. PAGE A underwent changes at times 7 and 13. Page B underwent changes at time 10, and so on.

In Figure 8, the arrows extending from the symbols "USER 1", etc., indicate the times of access by the users. For example, USER 1 accessed PAGE A, VERSION 1, at time 2. USER 1 then accessed PAGE A, VERSION 2, at time 9, and so on.

The invention maintains a TABLE of these times, as indicated on the right side of Figure 8, together with a list of PAGEs, or documents, owned by each USER. Ownership is determined by the hot lists. The invention also maintains (a) the BASE PAGES, (b) the changes to each, and (c) the times of each change, as indicated on the left side of the Figure. From this data, the invention is able to reconstruct any PAGE, as of any date subsequent to the date of the BASE PAGE.

Additional Considerations

1. One definition of "page" is that it refers to a unit of data, stored in a system, which is identified by a specific name. (In the WWW, all pages have unique names.) Other terms can refer to such units of data, such as "files" and "documents." In general, the particular name used will depend on the system storing the data.

2. One definition of "repository" is a collection of data, which is accessible by computer. The repository may be available to the public, or access may be limited. In general, repositories are expected to be distributed, meaning that the storage locations are physically distributed over a wide geographic area, and linked

together by a communication system.

3. It was stated above that the invention can reconstruct a page as of any selected date. The reconstruction is based on the changes 6 in Figure 3. These changes are detected periodically, and the periodicity is determined by each user of the system, subject to limits imposed by the designer and system administrator.

For example, user A can specify a period of one day for checking for changes in the pages on user A's hot list; user B can specify a different period for B's pages, such as one week. The system administrator can specify that no period, for any user, can be shorter than one hour.

Consequently, changes in a page, located in the REPOSITORY, will only appear in a reconstruction done by the EXTERNAL SERVICE after the changes have been detected, and not earlier. An example will illustrate this distinction.

Assume that the invention looks for changes on odd-numbered dates. Thus, a change occurring on the fourth of a month will be detected on the fifth. However, if a user happens to call for reconstruction on the fourth, the change occurring on the fourth will not appear in the reconstruction. Only changes occurring as of the prior detection, namely, as of the third, will appear.

It is expected that the detection process will be performed sufficiently often that the influence of this factor will be negligible.

4. The invention can extend its differencing function (ie, the examination of pages for changes) to pages referenced by the page accessed by the user. For example, if the user accesses document D in Figure 1B, the invention can detect changes in all documents referenced by document D, such as D1, D2, and D3.

In another embodiment, the differencing can extend to the documents which are, in turn, referenced by the referenced documents. For example, the referenced documents (D1, D2, and D3) refer to D5 and D6. These latter documents (D5 and D6) can be differenced also, as can be the documents which they reference, and so on.

5. The invention provides all information from which a current version of a PAGE may be derived. Figure 4 gives an example. Figure 4 contains all such information, together with other information which indicates changes since a previous version.

6. The discussion above presumed that comparison, or differencing, between different versions of a PAGE was done within the EXTERNAL SERVICE. This is not strictly necessary; the comparison can be done at any convenient location. Further, the preliminary checking for the existence of changes can be done at any convenient location.

7. In data storage systems, names are given to the units of information (e.g., documents, pages, records), although the names

can be different in different databases. However, the names of the units, in general, remain the same throughout time, despite changes which are made to the information contained in the unit. Therefore, one definition of the term "version" refers to a unit of information, which is different from a previous unit of the same name.

8. The REPOSITORY in Figure 1 is, in general, located remotely from the EXTERNAL SERVICE. Communication is undertaken by any convenient approach, such as a public-access communication network known as the INTERNET.

In general, the REPOSITORY is under independent control of the EXTERNAL SERVICE. One ramification of this independent control is that the type of processing done to the PAGEs copied into the EXTERNAL SERVICE is controlled by the EXTERNAL SERVICE, and not by the REPOSITORY. For example, (a) the particular processes used in locating and storing differences, (b) the frequency of processing, and (c) the mode of notifying a user, are controlled by the designer of the EXTERNAL SERVICE. The operator of the REPOSITORY has no involvement in this processing.

9. Figure 9 illustrates another form of the invention. The invention maintains base pages 30 within the EXTERNAL SERVICE, as required by the hot lists 36. The base pages 30 were downloaded from respective repositories 42A, 42B, etc.

The invention periodically monitors the originals 30A of the

pages, located in the repository 42, for changes, and stores the changes within the EXTERNAL SERVICE. The invention notifies users when changes are found in pages on their hot lists (notification is not shown).

A version control system 39 allows users to fetch and view any version of any page.

10. The different versions of documents may contain drawings, files from which sound may be generated, files which produce video clips and animation, and other components which do not consist strictly of alphanumeric characters. The invention detects the existence of changes in such components, and marks the existence of the changes, in the display as shown in FIgure 4, without necessarily identifying in detail the nature of the changes.

11. A primary use of the invention is envisioned in the situation shown in Figure 10. The EXTERNAL SERVICE obtains copies of PAGEs from a REPOSITORY, such as WWW. However, the EXTERNAL SERVICE is given no authority to replace or modify the pages contained in the REPOSITORY. To the EXTERNAL SERVICE, the PAGEs represent read-only data, as indicated by the "X" over arrow 50, which indicates a write operation.

The EXTERNAL SERVICE performs differencing between currently copied versions of pages, and DATA representing previous versions. The DATA stored in the EXTERNAL SERVICE can be both read, and written to, by the EXTERNAL SERVICE. The EXTERNAL SERVICE

reconstructs any version on demand, and also indicates differences between any two versions selected by a user, as discussed above. These functions can be accomplished by a prior-art Revision Control System, RCS (also called a Version Control System), or by the code contained in the listing contained in this Specification.

12. In one form of the invention, the PAGEs retrieved are written in a "markup language," such as HyperText Mark-up Language (HTML). A mark-up language, in general, contains two types of codes, interspersed among the actual text of a document.

One type indicates how the PAGEs are to be displayed. For example, some codes indicate paragraph indentation, other codes indicate font styles, yet other codes indicate style of font, within a font, such as italicizing, underlining, double-striking, or bold printing. This type of code is referred to as format-defining.

A second type of code can identify an image, such as a bit-mapped file located elsewhere. When such a code is read by the system displaying the PAGE, a copy of the image is retrieved, and displayed within the PAGE, at the location specified by the code. This type of code is referred to as content-defining.

The invention does not treat changes in the format-defining codes as changes in content. Thus, a PAGE which changes in layout, or typestyle, only, is not designated as a changed page.

The differencing program contained in the COMPUTER PROGRAM LISTING compares different versions on a subunit-by-subunit basis.

For example, the program compares corresponding sentences in different versions, and the sentences are detected by sentence terminators. (Longer subunits can be used, such as paragraphs or pages.) The sentence terminators are a subset of the markup language. Specifically, the terminators are format-defining codes.

COMPUTER PROGRAM LISTING

The program listing is divided into three sections.

1. HTMLDIFF, comprising:
   - -- html_diff.sml (5 pages),
   - -- diff.sml (3 pages),
   - -- mlweb.sml (4 pages), and
   - -- html.lex (one page).

2. W3NEWER (17 pages).

3. NOHANDS, comprising:
   - -- nohandsBE (11 pages),
   - -- no-hands.cgi (3 pages),
   - -- rcsdiff.cgo (4 pages), and
   - -- snapshot.cgi (3 pages).

NOHANDS is an overall program set which utilizes W3NEWER and HTMLDIFF.

TECHNICAL APPENDIX

A TECHNICAL APPENDIX, totalling 12 pages, two of which are

blank, follows, and refers to Figures 11 - 14.

Numerous substitutions and modifications can be undertaken without departing from the true spirit and scope of the invention. What is desired to be secured by Letters Patent is the invention as defined in the following claims.

CLAIMS

1. A system, comprising:

a) means for copying versions of pages from an external repository, which provides pages to the system on a read-only basis;

b) means for storing data from which

i) selected versions can be reconstructed; and

ii) differences between selected versions can be identified.

2. A system according to claim 1, in which the versions reconstructed are specified on lists provided by users.

3. In a system for allowing multiple users to interface with another system which contains versions of data, at least some of which change as time progresses, the improvement comprising:

a) means for allowing a user to identify a current version and a previous version; and

b) means for displaying the current version in a format which distinguishes the current version from the previous version.

4. A method of communicating with a repository which stores

data units in the form of current versions which change over time, comprising:

a) accepting lists of data units from multiple users;

b) maintaining an archive, which contains material which allows reconstruction of selected versions of data units contained on the lists; and

c) at intervals, comparing current versions with archived material, and updating the archive.

5. Method according to claim 4, and further comprising the step of

d) recording which versions each user has accessed.

6. Method according to claim 5, and further comprising the steps of:

e) accepting a request from a user for a data unit;

f) presenting a current version of the data unit; and

g) highlighting material within the current version which was absent from previous versions accessed by the user.

7. A method of communicating with a repository which stores data units in the form of versions which change over time, comprising:

a) maintaining an archive, which includes material which allows reconstruction of selected versions of data units;

b) accepting lists of data units from users;

c) for each list accepted, checking the archive and, if listed data units are not archived, copying a current version from the repository;

d) at intervals,

i) finding differences between (A) versions contained in the archive, and (B) current versions, contained in the repository, and

ii) storing the differences in the archive;

e) maintaining information about each user, which indicates most recent versions accessed by the user; and

f) accepting a request from a user for a current version and, in response,

i) copying the current version from the repository;

ii) finding differences between the current version and the most recent versions accessed; and

iii) displaying the current version in a format which highlights the differences.

8. Apparatus for comparing a first text written in a markup language, which includes sequences of words and markup commands, with a second such text comprising:

a) means for comparing the first and second text, which compares sequences in the first and second text which

are defined by a group of terminators, including sentence-ending punctuation and markup commands, belonging to a first subset thereof, and produces a result which indicates differences between the texts; and

b) means for receiving the result and displaying the differences in response thereto.

9. Apparatus for comparing versions of pages in a repository, comprising:

a) means for detecting that a second page is a new version of a first page;

b) means responsive to the detecting means for comparing the first page with the second page to produce a result which indicates differences between the first and second pages; and

c) means responsive to the result for displaying the differences.

10. A method for detecting whether a page in a repository has been modified since it was last examined by a given user, the method comprising the steps performed in the computer system used by the given user of:

a) maintaining a first record of last viewed times for pages in that computer system which indicates, for each page in a first set of pages, the last time at which the given user viewed the page and for a given page for which

a last viewed time is recorded in the first record, performing the steps of obtaining a last modified time for the given page from a second record in that computer system of last modified times of pages and determining whether the last modified time from the second record is a time such that the time is later than the last viewed time for the given record;

b) if the second record does not provide a last modified time which is such a time, obtaining a last modified time from a source external to that computer system to which the user has access and determining whether the last modified time from the external source is such a time; and

c) if the last modified time obtained from the external source is such a time, updating the second record with that last modified time.

11. A method for maintaining a user-defined version history of a page, the method comprising the steps of:

a) receiving an indication from the user specifying that a given version of a page is to be retained in a version history for that page;

b) if the given version is the first version of the page in the version history, storing a copy of the given version in the version history; and

c) otherwise storing at least any difference between the

given version and the most recent previous version in the version history.

TECHNICAL APPENDIX

Tracking and Viewing Changes on the Web

Abstract

We describe a set of tools that detect when World-Wide-Web pages have been modified and present the modifications visually to the user through marked-up HTML. The tools consist of three components: *w3newer*, which detects changes to pages; *snapshot*, which permits a user to store a copy of an arbitrary Web page and to compare any subsequent version of a page with the saved version; and *htmldiff*, which marks up HTML text to indicate how it has changed from a previous version. We refer to the tools collectively as the *Network-Oriented HTML Archival, Notification, and Differencing System* (NO HANDS). This paper discusses several aspects of NO HANDS, with an emphasis on systems issues such as scalability, security, and error conditions.

1 Introduction

Use of the World-Wide-Web (W^3) has increased dramatically over the past couple of years, both in the volume of traffic and the variety of users and content providers. The W^3 has become an information distribution medium for academic environments (its original motivation), commercial ones, and virtual communities of people who share interests in a wide variety of topics. Information that used to be sent out over electronic mail or USENET, both active media that go to users who have subscribed to mailing lists or newsgroups, can now be posted on a W^3 page. Users interested in that data then visit the page to get the new information.

The URLs of pages of interest to a user can be saved in a "hotlist" (known as a bookmark file in Netscape™), so they can be visited conveniently. How does a user find out when pages have changed? If users know that pages contain up-to-the-minute data (such as stock quotes), or are frequently changed by their owners, they may visit the pages often. Other pages may be ignored, or browsed by the user only to find they have not changed.

In recent months, several tools have become available to address the problem of determining when a page has changed. One example of such a tool is *webwatch*, a product for Windows™ that uses the HTTP HEAD command to find out when a page has been modified since it was last viewed by a user's web browser, and generates a report in HTML that allows the user to go directly to those updated pages. Another example is *w3new*, by Brooks Cutter, a public-domain *perl* script that runs on UNIX® [2].

Each of these tools suffers from a significant deficiency: while they provide the user with the knowledge that the page has changed, they do not show *how* the page has changed. Although a few pages are edited by their maintainers to highlight the most recent changes, often the modifications are not prominent, especially if the pages are large. Even pages with special highlighting of recent changes are problematic: if a user visits a page frequently, what is "new" to the maintainer may not be "new" to the user. Alternatively, a user who visits a page infrequently may miss changes that the maintainer deems to be old.

We have developed a system that efficiently tracks when pages change, compactly stores versions on a per-user basis, and automatically compares and presents the differences between pages. NO HANDS (*Network-Oriented HTML Archival, Notification, and Differencing System*) provides "personalized" views of versions of W^3 pages with three tools. The first, *w3newer*, is a more scalable version of Cutter's *w3new* modification tracking tool that periodically accesses the W^3 to find when pages on a user's hotlist have changed. The second, *snapshot*, allows a user to save versions of a page and later use a third tool, *htmldiff*, to see how it has changed. *Htmldiff* automatically compares two HTML pages and creates a "merged" page to show the differences with special HTML markups.

™ Netscape is a trademark of Netscape Communications Corporation.

™ Windows is a trademark of Microsoft.

® UNIX is a registered trademark of X/Open.

checked. The threshold can vary depending on the URL, with *perl* pattern matching used to determine what threshold to apply. The first matching pattern is used. Table 1 gives an example of a "/.w3newer_thresholds configuration file. Thresholds are specified as combinations of days (d) and hours (h), with 0 indicating that a page should be checked on every run of *w3newer* and never indicating that it should never be checked.

# Comments start with a sharp sign.	
# perl syntax requires that "." be escaped	
# Default is equivalent to ending the file with ".*"	
Default	2d
file:.*	0
http://www\.yahoo\.com/.*	7d
http://www\.research\.att\.com/.*	0
http://.*\.att\.com/.*	1h
http://home\.mcom\.com/home/whatsnew/-whats_new\.html	12h
http://www\.ncsa\.uiuc\.edu/SDG/Software/-Mosaic/Docs/whats-new\.ht	12h
http://snapple\.cs\.washingto\.edu:600/-mobile/	1d
# rarely modified	
http://www\.cs\.duke\.edu/~pk/-HomePage\.html	7d
# this is in my hotlist but will be different every day	
http://www\.unitedmedia\.com/-comics/dilbert/	never

Table 1: An example of the thresholds specified to w3newer.

2.3 Cache Consistency Issues

Determining when HTTP pages have changed is analogous to caching a file in a distributed file system and determining when the file has been modified. While file systems such as the Andrew File System [6] and Sprite [8] provide guarantees of cache consistency by issuing call-backs to hosts with invalid copies, HTTP access is closer to the traditional NFS [12] approach, in which clients check back with servers periodically for each file they access. Netscape can be configured to check the modification date of a cached page each time it is visited, once each session, or not at all. Caching servers check when a client forces a full reload, or after a time-to-live value expires.

Here the problem is complicated by the target environment: one wishes to know not only when a currently viewed page has changes, but also when a page that has not been seen in a while has changed. Fortunately, unlike with file systems, HTTP data can usually tolerate some inconsistency. In the case of pages that are of interest to a user but have not been seen recently, finding out within some reasonable period of time, such as a day or a week, will usually suffice. Even if servers had a mechanism to notify all interested parties when a page has changed, immediate notification might not be worth the overhead.

Instead, one could envision using something like the Harvest replication and caching services [1] to notify interested parties in a lazy fashion. A user who expresses an interest in a page, or a browser that is currently caching a page, could register an interest in the page with its local caching service. The caching service would in turn register an interest with an Internet-wide, distributed service that would make a best effort to notify the caching service of changes in a timely fashion. (This service could potentially archive versions of HTTP pages as well.) Pages would already be replicated, with server load distributed, and the mechanism for discovering when a page changes could be left to a negotiation between the distributed repository and the content provider: either the content provider notifies the repository of changes, or the repository polls it periodically. Either way, there would not be a large number of clients polling each interesting HTTP server. Moving intelligence about HTTP caching to the server has been proposed by Gwertzman and Seltzer [3] and others.

One could also envision integrating the functionality of NO HANDS into file systems. Tools that can take actions when arbitrary files change are not widely available, though they do exist [11]. Users might like to have a unified report of new files and W^3 pages, and *w3newer* supports the "file:" specification and can find out if a local file has changed. However, *snapshot* has no way to access a file on the user's (remote) file system. Moving functionality into the browser would allow individual users to take snapshots of files that are not already under the control of a versioning system such as the Revision Control System (RCS) [14]; this might be an appropriate use of a browser with client-side execution, such as *HotJava* [13].

2.4 Error Conditions

When a periodic task checks the status of a large number of URLs, a number of things can go wrong. Local problems such as network connectivity or the status of a proxy-caching server can cause *all* HTTP requests to fail. Proxy-caching servers are sometimes overloaded to the point of timing out large numbers of requests, and a background task that retrieves many URLs in a short time can aggravate their condition. *W3newer* should therefore be able to detect cases when it should abort and try again later (preferably in time for the user to see an updated report).

At the same time, a number of errors can arise with indi-

vidual URLs. They can move, with or without leaving a forwarding pointer. The server for a URL can be deactivated or renamed. They may disallow retrieval by "robots," meaning that any program that follows the "robot exclusion protocol" [10] will not retrieve them. Since the cost of retrieving modification dates is small in comparison to the cost of retrieving robots.txt (part of the exclusion protocol), it may well be appropriate to ignore the robot exclusion protocol for this task, or to check robots.txt only occasionally on each host. Observing the protocol will still be advisable for hosts on which many URLs are checked, especially if the pages' contents are retrieved each time.

Finally, automatic detection of modifications based on information such as modification date and checksum can lead to the generation of "junk mail" as "noisy" modifications trigger change notifications. For instance, pages that report the number of times they have been accessed, or embed the current time, will look different every time they are retrieved.

W3newer attempts to address these issues by the following steps:

- If a URL is inaccessible to robots, that fact is cached so the page is not accessed again unless a special flag is set when the script is invoked.
- Another flag can tell *w3newer* to treat error conditions as a successful check as far as the URL's timestamp goes. For instance, if *w3newer* runs daily and checks a particular URL every four days, normally an error accessing the page on Monday will cause it to be checked again on Tuesday. With this flag, it would be checked again on Friday. In general, it seems that errors are likely to be transient, and checking the next time *w3newer* is run would be reasonable.
- When a URL is inaccessible, an error message appears in the status report, so the user can take action to remove a URL that no longer exists or repeatedly hits errors.

In addition, *w3newer* could be modified to keep a running counter of the number of times an error is encountered for a particular URL, or to skip subsequent URLs for a host if a host or network error (such as "timeout" or "network unreachable") has already occurred. Addressing the problem of "noisy" modifications will require heuristics to examine the differences at a semantic level.

3 Snapshots: External Representations of Version Histories

In addition to providing a mechanism for determining when W^3 pages have been modified, there must be a way to access multiple versions of a page for the purposes of comparison. This section describes methods for maintaining version histories and several issues that arise with our solution.

3.1 Alternative Approaches

There are three possible approaches for providing versioning of W^3 pages: making each content provider keep a history of all versions, making each user keep this history, or storing the version histories on an external server.

Server-side support Each server could store a history of its pages and provide a mechanism to use that history to produce marked-up pages that highlight changes. This method requires arbitrary content providers to provide versioning and differencing, so it is not practical, although it is desirable to support this feature when the content provider is willing. (See Section 6.1.)

Client-side support Each user could run a program that would store items in the hotlist locally, and run *htmldiff* against a locally saved copy. This method requires that every page of interest be saved by every user, which is unattractive as the number of pages in the average user's hotlist increases, and it also requires the ability to run *htmldiff* on every platform that runs a W^3 browser. Storing the pages referenced by the hotlist may not be too unreasonable, since programs like Netscape may cache pages locally anyway. There are other external tools such as *warmlist* [16] that provide this functionality.

External service Our approach is to run a service that is separate from both the content provider and the client. Pages can be registered with the service via an HTML form, and differences can be retrieved in the same fashion. Once a page is stored with the service, subsequent requests to remember the state of the page result in an RCS "check-in" operation that saves only the differences between the page and its previously checked-in version. Thus, except for pages that change in many respects at once, the storage overhead is minimal beyond the need to save a copy of the page in the first place.

Drawbacks to the "external service" approach are that the service must remember the state of every page that anyone who uses the service has indicated an interest in, and must know which user has seen which version of each page. The first issue is primarily one of resource allocation, and is not expected to be a significant issue unless the service is used by a great many clients on a number of large pages. The second issue is addressed by using RCS's support for datestamps and requesting a page as it existed at a particular

time. Alternatively, a version number could be retained for each <user,URL> combination.

Relative links become a problem when a page is moved away from the machine that originally provided it. If the source were passed along unmodified, then the W^3 browser would consider links to be relative to the CGI directory containing the *snapshot* script. HTML supports a BASE directive that makes relative links relative to a different URL, which mostly addresses this problem; however, Netscape 1.1N treats internal links within such a document to be relative to the new BASE as well, which can cause the browser to jump between the *htmldiff* output and the original document unexpectedly.

3.2 System Issues

The *snapshot* facility must address four important issues: use of CGI, synchronization, resource utilization, and security/privacy.

CGI is a problem because there is no way for *snapshot* to interact with the user and the user's browser, other than by sending HTML output. When a CGI script is invoked, *httpd* sets up a default timeout, and if the script does not generate output for a full timeout interval, *httpd* will return an error to the browser. This was a problem for *snapshot* because the script might have to retrieve a page over the Internet and then do a time-consuming comparison against an archived version. The server does not tell *snapshot* what a reasonable timeout interval might be for any subsequent retrievals; instead this is hard-coded into the script. In order to keep the HTTP connection alive, *snapshot* forks a child process that generates one space character (ignored by the W^3 browser) every several seconds while the parent is retrieving a page or executing *htmldiff*.

Synchronization between simultaneous users of the facility is complicated by the use of multiple files for bookkeeping. The system must synchronize access to the RCS repository, the locally cached copy of the HTML document, and the control files that record which version of each page a user has seen. Currently this is done by using UNIX file locking on both a per-URL lock file and the per-user control file. Ideally the locks could be queued such that if multiple users request the same page simultaneously, the second *snapshot* process would just wait for the page and then return, rather than repeating the work. This is not so important for making snapshots, in which case a proxy-caching server can respond to the second request quickly and RCS can easily determine that nothing has changed, but there is no reason to run *htmldiff* twice on the same data.

The latter point relates to the general issue of resource utilization. *Snapshot* has the potential to use large amounts of both processing and disk space. The need to execute *htmldiff* on the server can result in high processor loads if the facility is heavily used. These loads can be alleviated by caching the output of *htmldiff* for a while, so many users who have seen versions N and $N-1$ of a page could retrieve *htmldiff*($page_N$, $page_{N-1}$) with a single invocation of *htmldiff*. The facility could also impose a limit on the number of simultaneous users, or replicate itself among multiple computers, as many W^3 services.

Disk space is potentially a problem if the repository can grow without bound and with no cost to its users. In fact, before a service like this could be placed on the Internet, it would have to authenticate each user and limit the user to a fixed number of URLs and/or disk blocks. Most likely, one would use an Internet commerce facility to charge a fee in exchange for permission to store a collection of URLs; this fee could easily offset the cost of the storage medium since it would also be paying for the differencing service.

Lastly, security and privacy are important. Because the CGI scripts run with minimal privileges, from an account to which many people have access, the data in the repository is vulnerable to any CGI script and any user with access to the CGI area. Data in this repository can be browsed, altered, or deleted. In order to use the facility one must give an identifier (currently one's email address, which anyone can specify) that is used subsequently to compare version numbers. Browsing the repository can therefore indicate which user has an interest in which page, how often the user has saved a new checkpoint, and so on.

By moving to an authenticated system on a secure machine, one could break some of these connections and obscure individuals' activities while providing better security. The repository would associate impersonal account identifiers with a set of URLs and version numbers, and passwords would be needed to access one of these accounts. Whoever administers this facility, however, will still have information about which user accesses which pages, unless the account creation can be done anonymously.

4 HtmlDiff: Comparison of HTML pages

In our experience, only a small fraction of pages on the W^3 contain information that allows users to ascertain how the pages have changed—examples include icons that highlight recent additions, a link to a "changelog", or a special "What's New" page. As was mentioned in the introduction, these approaches suffer from deficiencies. They are intended to be viewed by all users, but users will visit the pages at different intervals and have different ideas of "what's new". In addition, the maintainer must explicitly generate the list of recent changes, usually by manually

marking up the HTML.

Automatic comparison of HTML pages and generation of marked-up pages frees the HTML provider from having to determine what's new and creating new or modified HTML pages to point to the differences. There are many ways to compare documents and many ways to present the results. This section describes various models for the comparison of HTML documents, our comparison algorithm, and issues involved in presenting the results of the comparison.

4.1 What's in a Diff?

HTML separates content (raw text) from markups. While many markups (such as <P>, <I>, and <HR>) simply change the formatting and presentation of the raw text, certain markups such as images (<IMG src=...>) and hypertext references (<A href= ...>) are "content-defining." Whitespace in a document does not provide any content (except perhaps inside a <PRE>), and should not impact comparison.

At one extreme, one can view an HTML document as merely a sequence of words and "content-defining" markups. Markups that are not "content-defining" as well as whitespace are ignored for the purposes of comparison. The fact that the text inside <P>...</P> is logically grouped together as a paragraph is lost. As a result, if one took the text of a paragraph comprised of four sentences and turned it into a list (<UL>) of four sentences (each starting with <LI>), no difference would be flagged because the content matches exactly.

At the other extreme, one can view HTML as a hierarchical document and compare the parse tree or abstract syntax tree representations of the documents, using subtree equality (or some weaker measure) as a basis for comparison. In this case, a subtree representing a paragraph (<P>...</P>) might be incomparable with a subtree representing a list (<UL>...</UL>). The example of replacing a paragraph with a list would be flagged as both a content and format change.

We view an HTML document as a sequence of sentences and "sentence-breaking" markups (such as <P>, <HR>, <LI>, or <H1>) where a "sentence" is a sequence of words and certain (non-sentence-breaking) markups (such as <B> or <A>). A "sentence" contains at most one English sentence, but may be a fragment of an English sentence. All markups are represented and are compared, regardless of whether or not those markups are "content-defining." In the paragraph-to-list example, the comparison would show no change to content, but a change to the formatting.

We apply Hirshberg's solution to the longest common subsequence (LCS) problem [4, 5] (with several speed optimizations) to compare HTML documents. This is the well-known comparison algorithm used by the Unix *diff* utility [7]. The LCS problem is to find a (not necessarily contiguous) common subsequence of two sequences of tokens that has the longest length (or greatest weight). Tokens not in the LCS represent changes. In Unix *diff*, a token is a textual line and each line has weight equal to 1. In *htmldiff*, a token is either a sentence-breaking markup or a sentence, which consists of a sequence of words and non-sentence-breaking markups. Note that the definition of sentence is *not* recursive; sentences cannot contain sentences. A simple lexical analysis of an HTML document creates the token sequence and converts the case of the markup name and associated (variable,value) pairs to upper-case; parsing is not required.

We now describe how the weighted LCS algorithm compares two tokens and computes a non-negative weight reflecting the degree to which they match (a weight of 0 denotes no match). Sentence-breaking markups can only match sentence-breaking markups. They must be identical (modulo whitespace, case, and reordering of (variable,value) pairs) in order to match (see section 4.3 for a discussion of the ramifications of this). A match has weight equal to 1. Sentences can match only sentences, but sentences need not be identical to match one another. We use two steps to determine whether or not two sentences match. The first step uses sentence length as a comparison metric. Sentence length is defined to be the number of words and "content-defining" markups such as <IMG> or <A> in a sentence. Markups such as <B> or <I> are not counted. If the lengths of two sentences are not "sufficiently close," then they do not match. Otherwise, the second step computes the LCS of the two sentences (where words matching exactly against words are assigned weight 1, and markups match exactly against markups, as before). Let W be the number of words and content-defining markups in the LCS of the two sentences and let L be the sum of the lengths of the two sentences. If the percentage $(2 * W)/L$ is sufficiently large, then the sentences match with weight W. Otherwise, they do not match.

4.2 Presentation of the differences

The comparison algorithm outlined above yields a mapping from the tokens of the old document to the tokens of the new document. Tokens that have a mapping are termed "common"; tokens that are in the old (new) document but have no counterpart in the new (old) are "old" ("new"). We refer to the "old" and "new" tokens as "differences".

We investigated three basic ways to present the differences by creating HTML documents that highlight the dif-

ferences with a variety of markup techniques:

Side-by-Side A side-by-side presentation of the documents with common text vertically synchronized is a very popular and pleasing way to display the differences between documents (see, for example, Unix *sdiff* or SGI's graphical diff tool *gdiff*). Unfortunately, there is no good mechanism in place with current HTML and browser technology that allows such synchronization (although it might be possible to make a document that contained a table with a document per column in which rows of the table were used to achieve synchronization).

Only Differences Show only differences (old and new) and eliminate the common part (as done in Unix *diff*). This optimizes for the "common" case, where there is much in common between the documents. This is especially useful for very large documents but can be confusing because of the loss of surrounding common context. Another problem with this approach is that an HTML document comprised of an interleaving of old and new fragments might be syntactically incorrect.

Merged-page Create an HTML page that summarizes all of the common, new, and old material. This has the advantage that the common material is displayed just once (unlike the side-by-side presentation). However, incorporating two pages into one again raises the danger of creating syntactically or semantically incorrect HTML (consider converting a list of items into a table, for example).

Our preference is to present the differences in the merged-page format to provide context and use internal hypertext references to link the differences together in a chain so the user can quickly jump from difference to difference. We currently deal with the syntactic/semantic problem of merging by eliminating all old markups from the merged page (note that this doesn't mean all markups in the older document, just the ones classified as "old" by the comparison algorithm). As a result, old hypertext references and images do not appear in the merged page (of course, since they were deleted they may not be accessible anyway). However, by reversing the sense of "old" and "new" one can create a merged page with the old markups intact and the new deleted. A more Draconian option would be to leave out all old material. In this case, there are no syntactic problems given that the most recent page is syntactically correct to begin with; the merged page is simply the most recent page plus some markups to point to the new material. We are exploring other ways to create a merged page.

An example of *htmldiff*'s merged-page output appears in Figure 1. Markups are used to highlight old and new material as follows. Two small arrow images are used to point to areas in the document that have changed. A red arrow points to old content and a green arrow points to new content. The arrows are also internal hypertext references to one another, linked in a chain to allow quick traversal of the differences. A banner at the front of the document contains a link to the first difference. Old text is displayed in "struck-out" font using <STRIKE>, which in our experience is rarely used in HTML found on the W[3]. Unfortunately, there is no ideal font for showing "new" text. We currently use <STRONG><I>. Ideally, we would like to be able to color code the text or text background to highlight old and new text, but this capability is not provided by current browsers. Another approach would be to choose a font that is not active at the point of the difference.

Note that not all changes in the documents are highlighted. For example, new markups that are not "content-defining" (such as <P>) are not marked up. However, markups such as anchors are highlighted. Consider the example of changing the URL in an anchor but not the content surrounded by <A>...</A>. In this case, an arrow will point to the text of the anchor, but the text itself will be in its original font, signifying a change to just the URL.

4.3 Issues and Extensions

Since *htmldiff* can parse an HTML document and rectify certain syntactic problems, such as mismatched or missing markups, the only real problem it is likely to encounter is a set of changes that are so pervasive as to make the resulting merged HTML unreadable. For instance, if every other line were changed, then the mixture of unrelated struck-out and emphasized text would be muddled. We are experimenting with methods for varying the degree to which old and new text can be interspersed, as well as thresholds to specify when the changes are too numerous to display meaningfully.

Currently, *htmldiff* is neither "version-aware" nor "web-aware". That is, *htmldiff* only compares the text of two HTML pages. It does not compare versions of the entities that the pages refer to, access them, or invoke itself recursively on other referenced pages. This has a number of consequences. The good news is that *htmldiff* does not incur the overhead of pulling versions from a repository or sending requests over the W[3] for information. This cost is consumed by *w3newer* and *snapshot*. The bad news is that some differences may be ignored. For example, if the contents of an image file are changed but the URL of the file does not, then the URL in the page will not be flagged as changed. To support such comparison would require some sort of versioning of referenced entities and would also require *htmldiff* to have access to the version repositories. Full versioning of all entities would allow interesting com-

BLANK PAGE

parisons to be done, but would dramatically increase storage requirements. A cheaper alternative would be to store a checksum of each entity and use the checksums to determine if something has changed. We are exploring how to efficiently perform such "smarter" comparisons.

5 Integrating the tools

There are two entry points to NO HANDS, one through *w3newer* and one through *snapshot*.

Currently, *w3newer* is invoked directly by the user, probably by a *crontab* entry, and generates an HTML document indicating which pages have changed. If specified, *w3newer* will associate three links with each document in the hotlist:

Remember Send the URL to the *snapshot* facility, to save a copy of the page. Though the page is retrieved, the RCS *ci* command ensures that it is not saved if it is unchanged from the previous time it was stored away.

Diff Have the *snapshot* facility invoke *htmldiff* to display the changes in a page since it was last saved away by the user.

History Have *snapshot* display a full log of versions of this page, with the ability to run *htmldiff* on any pair of versions or to view a particular version directly. (See Figure 2.)

Thus, each page that is reported as "new" can immediately be passed to *htmldiff*, and any page in the list can be "remembered" for future use. An example of *w3newer*'s output appears in Figure 3.

A user may also to choose to enter *snapshot* directly to check-in pages, or view the current page or the version history. Figure 4 shows the interface to NO HANDS through *snapshot*. If the user selects the *history* link, the page shown in Figure 2 is presented. Finally, selecting two pages to compare invokes *htmldiff*, as in Figure 1.

One disadvantage of the current approach is that there is no direct interaction between *w3newer*, *snapshot*, and the W^3 browser. Viewing a page with *htmldiff* does not cause the browser to record that the page has just been seen; instead, the browser records the URL that was used to invoke *htmldiff* in the first place. Subsequently, *w3newer* uses the obsolete datestamp from the browser and continues to report that the page has been modified more recently than the browser has seen it. As a result, the user must view a page directly as well as via *htmldiff* in order to both remove it from the list of modified pages and see the actual differences

6 Extensions

This section describes some possible extensions to the work already presented. Section 6.1 discusses an interface between RCS and *htmldiff* that is already implemented, while Sections 6.2 and 6.3 presents unimplemented extensions to integrate tracking modifications into the server and to invoke scripts via the HTTP POST protocol.

6.1 Server-side Version Control

The tools described above do not require any changes to arbitrary servers or clients on the W^3. Existing GET and POST protocols are used to communicate with specific servers that save versions of documents and provide marked-up versions showing how they have changed. However, if a server runs *htmldiff* and some *perl* scripts, it can provide a direct version-control interface and avoid the need to store copies of its HTML documents elsewhere.

The *perl* scripts we have written provide an interface to RCS [14]. A CGI script (/cgi-bin/rlog) converts the output of *rlog* into HTML, showing the user a history of the document with links to view any specific version or to see the differences between two versions. Another script (/cgi-bin/co) displays a version of a document under RCS control, while still another (/cgi-bin/resdiff) displays the differences. If the file's name ends in .html then *htmldiff* is used to display the differences, rather than the *resdiff* program.

As an example, one might set up a Last-Modified field at the bottom of an HTML document to be a link to the rlog script, with the document name specified as a parameter. After clicking on this unobtrusive field, the user would be able to see the history of the document.

6.2 Server-side URL Tracking

Currently, *w3newer* runs on the user's machine, so multiple instantiations of the script may perform the same work. Although it runs a related daemon on the same machine as an AT&T-wide proxy-caching server, which returns information about pages that are currently cached on the server and may eliminate some accesses over the Internet, there is insufficient locality in that cache for it to eliminate a significant fraction of requests.

Alternatively, *w3newer* could be run on the set of pages that have been saved by the *snapshot* daemon. Regardless of how many users have registered an interest in a page, it need only be checked once: if changed, the new version could be saved automatically. Then a user could request a list of all pages that have been saved away, and get an indication of which pages have changed since they were saved by the user.

BLANK PAGE

Adding this functionality would be useful, since it would offer economies of scale. It would have the disadvantage of being decoupled from a given user's W^3 browser history; i.e., if a user views a page directly, the *snapshot* facility would have no indication of this and might present the page as having been modified.

6.3 Interactions with CGI scripts

Because NO HANDS can handle arbitrary URLs, it can interact with CGI scripts that use the GET protocol by passing arguments to the script as part of the URL. However, services that use POST cannot be accessed, because the input to the services is not stored.

Both *w3newer* and *snapshot* would have to be modified to support the POST protocol, in order to invoke a service and see if the result has changed, and then to store away the result and display the changes if it has. The interface to NO HANDS to support POST is unclear, however. A user could manually save the source to an HTML form and change the URL the form invokes to be something provided by NO HANDS. It, in turn, would have to make a copy of its input to pass along to the actual service. The result would be an HTTP equivalent of a UNIX *pipe*, interposing an extra service between the browser and the service the user is trying to invoke.

Instead, the browser could be modified to have better support for forms:

- It should store the filled-out version of a form in its bookmark file, so the user could jump directly to the output of a CGI script.
- It should be able to pass a form directly to NO HANDS, along with the URL specified in the FORM tag, so that the output could be stored under RCS.

7 Conclusions

NO HANDS combines notification, archiving, and differencing of W^3 pages into a single cohesive tool. It achieves economies of scale by avoiding unnecessary HTTP accesses, saving pages at most once each time they are modified (regardless of the number of users who track it), and using RCS as the underlying versioning system. Automatic generation of differences within the HTML framework provides users with the ability to see both insertions and deletions in a convenient fashion.

In the general setting of the W^3 and document retrieval, NO HANDS benefits two communities: users of the W^3 no

longer have to browse to find pages of interest that have changed; HTML providers no longer have to create suitably marked-up pages to show "what's new". While such automation is clearly helpful in this general context, we expect that NO HANDS will be a critical part of more focused uses of the W^3, especially in areas involving collaborative and distributed work.

Several issues still need to be addressed. In particular, many of the complications of NO HANDS could be avoided by better integration with W^3 browsers and servers. For instance, viewing the difference between an older version of a page and its current version should update the browser's notion of when the page was last visited. Finally, the increasing availability of distributed, hierarchical HTTP repositories such as Harvest [1] will be both an opportunity and a challenge for scalable notification mechanisms and version archives.

References

[1] C. Mic Bowman et al. Harvest: A scalable, customizable discovery and access system. Technical Report CU-CS-732-94, Dept. of Computer Science, University of Colorado–Boulder, March 1995.

[2] B. B. Cutter III. w3new. http://www.stuff.com/~bcutter/programs/w3new/w3new.html.

[3] James S. Gwertzman and Margo Seltzer. The case for geographical push-caching. In *Proceedings of the Fifth Workshop in Hot Topics in Operating Systems (HOTOS-V)*, pages 51–55, Orcas Island, WA, May 1995. IEEE.

[4] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–343, June 1975.

[5] D. S. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the ACM*, 24(4):664–675, October 1977.

[6] J. Howard et al. Scale and performance in a distributed file system. *ACM Transactions on Computer Systems*, 6(1):51–81, February 1988.

[7] J. W. Hunt and M. D. McIlroy. An algorithm for differential file comparison. Technical Report Computing Science TR #41, Bell Laboratories, Murray Hill, N.J., 1975.

[8] M. Nelson, B. Welch, and J. Ousterhout. Caching in the Sprite network file system. *ACM Transactions on Computer Systems*, 6(1):134–154, February 1988.

[9] M. Newbery. Kaupo. http://www.vuw.ac.nz/~newbery/Kaupo.html

[10] A standard for robot exclusion. http://web.nexor.co.uk/mak/doc/robots/norobots.html

[11] David S. Rosenblum and Balachander Krishnamurthy. Generalized event-action handling. In Balachander Krishnamurthy, editor, *Practical Reusable UNIX Software*, chapter 9. John Wiley & Sons, New York, 1995.

[12] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon. Design and implementation of the Sun network filesystem. In *Proceedings of the USENIX 1985 Summer Conference*, pages 119–130, June 1985.

[13] Sun Microsystems. *The HotJava Browser. A White Paper*. Available as http://java.sun.com/1.0alpha3.doc/overview/hotjava/browser-white paper.ps.

[14] W. Tichy. RCS: a system for version control. *Software—Practice & Experience*, 15(7):637–654, July 1985.

[15] Url-minder. http://www.netmind.com/URL-minder/URL-minder.html

[16] Warmlist. http://glimpse.cs.arizona.edu:1994/~paul/warmlist/

COMPUTER PROGRAM LISTING

```
structure Html diff =
    struct
      open StringUtil
      open Html

      datatype diff = SAME | NEW | OLD | NEWERASE

      fun is_visible (SEQ(l)|BEGEND(l)) =
          (List.exists (fn a => is_visible a) l)
        | is_visible (RAW(s)) = true
        | is_visible (MARKUP("IMG",_)) = true
        | is_visible (MARKUP("A",l)) =
          List.exists (fn(x,y)=> (x="HREF")) l
        | is_visible h = false

      fun has_text (SEQ(l)|BEGEND(l)) =
          (List.exists (fn x => has_text x) l)
        | has_text (RAW(_)) = true
        | has_text _ = false

      (* expression comparison *)
      structure HtmlOrder : ORD_KEY =
        struct
          type ord_key = string;
          fun cmpKey (s1,s2) = StringUtil.strcmp(s1,s2)
        end;

      (* sets of strings *)
      structure Hmap = SplayDict(HtmlOrder);

      (* diffing html docs *)
      structure HtmlWeight : WEIGHT =
        struct
          type T = HtmlDoc
          fun weight diff (h1,h2) =
            let
              fun r_length l =
                let
                  fun r_l (x::xs) c =
                      if is_visible x then
                        r_l xs (c+1)
                      else
                        r_l xs c
                    | r_l [] c = c
                in
                  r_l l 0
                end
              fun r_common l ind =
                let
                  fun r_c (x::xs) ((i,j,t)::ys) k c =
                      if i = k then
                        if is_visible x then
                          r_c xs ys (k+1) (c+1)
                        else
                          r_c xs ys (k+1) (c)
                      else
                        r_c xs ((i,j,t)::ys) (k+1) c
                    | r_c          = c
                in
                  r_c l ind 0 0
                end
              fun markup s1 s2 l1 l2 =
                DIFF(if s1 = s2 andalso l1 = l2
                       then 1 else 0,[])
              fun v ((RAW s1),(RAW s2)) =
                  if s1 = s2 then DIFF(1,[]) else DIFF(0,[])
                | v ((BEGEND(l1)),(BEGEND(l2))) =
                  let
                    val mi = real(min(r_length(l1),
                                      r_length(l2)))
                    val ma = real(max(r_length(l1),
                                      r_length(l2)))
                  in
                    if (ma = 0.0 orelse (mi/ma) < 0.6) then
                      DIFF(0,[])
                    else
                      let val DIFF(v,l) =
                          (diff
                           (Vector.vector l1)
                           (Vector.vector l2))
                          val rc = r_common l1 l
                      in
                        if (mi <> 0.0 andalso (real(rc))/mi > 0.8) then

                          DIFF(rc,l)
                        else
                          DIFF(0,[])
                      end
                  end
                | v ((m1 as (MARKUP(s1,l1))),
                     (m2 as (MARKUP(s2,l2)))) =
                    markup s1 s2 l1 l2
                | v ((m1 as (ENDMARKUP(s1,l1))),
                     (m2 as (ENDMARKUP(s2,l2)))) =
                    markup s1 s2 l1 l2
                | v ((m1 as (PROCMARKUP(s1,l1))),
                     (m2 as (PROCMARKUP(s2,l2)))) =
                    markup s1 s2 l1 l2
                | v ((m1 as (DECLMARKUP(s1,l1))),
                     (m2 as (DECLMARKUP(s2,l2)))) =
                    markup s1 s2 l1 l2
                | v (_,_) = DIFF(0,[])
            in
              v (h1,h2)
            end
        end

      structure HtmlDiff = DiffFUN(structure W = HtmlWeight)

      (* functions for dealing with whitespace in RAW *)
      fun eliminate_white h =
        let
          fun is_raw_white (RAW(s)) =
            let
              fun whitespace s =
                let
```

```
                    fun ww [#[#" "]|#["\n"]|#["\t"]] :: xs) = ww xs
                      | ww [] = false
                      | ww _ = true
                in
                    ww (explode s))
                end

            in
                whitespace s
            end
          | is_raw_white _ = false;

        fun html_white [x::y::xs] (res,wh) =
            if is_raw_white y then
                html_white xs [x::res,y::wh]
            else
                html_white [y::xs] [x::res,(RAW(""))::wh]
          | html_white [x::[]] (res,wh) =
            if is_raw_white x then
                (rev res,rev wh)
            else
                (rev [x::res],rev [(RAW(""))::wh])
          | html_white [] (r,w) = (rev r,rev w)
    in
        html_white h ([],[])
    end

fun merge_white m new_v =
    let
        fun upd h v n (res,nc) =
            (((SEQ([h,Vector.sub(v,n)]))::res),nc))

        fun rep_wh (SEQ([]|BEGEND(_)) t (res,nc) =
            let
                fun m2 [x::xs] r = (m2 xs (rep_wh x t r))
                  | m2 [] r = r
                val (res2,nc2) = (m2 l ([],nc))
            in
                (((SEQ(rev res2))::res),nc2)
            end
          | rep_wh h SAME (res,nc:int) =
            upd h new_v nc (res,nc+1)
          | rep_wh h (NEW(NEWERASE)) (res,nc) =
            upd h new_v nc (res,nc+1)
          | rep_wh h OLD (res,nc) =
            (((SEQ([h,RAW(" ")]))::res),nc)

        fun ml ([t,x]::xs) (res,nc) =
            let val (r,n) = (rep_wh x t ([],nc)) in
                ml ((t,(hd r))::(m1 xs ([],n)))
            end
          | ml [] r = []
    in
        (ml m ([],0))
    end

(* merge two html pages into one *)
fun html_merge h1 h2 new_only =
```

```
let
    fun gen_list [] [] _ _ _ =
        []
      | gen_list (x::xs) l _ _ [] =
        if new_only then
            []
        else
            (OLD,x)::(gen_list xs l 0 0 [])
      | gen_list [] (y::ys) _ _ [] =
            (NEW,y)::(gen_list [] ys 0 0 [])
      | gen_list l1 l2 p1 p2 (d1 as ((x,y,d2)::xs)) =
            if p1 = x andalso p2 = y then
                (case (hd l1) of
                    BEGEND(b1) =>
                        let
                            val DIFF(w,d) = d2
                            val BEGEND(b2) = (hd l2)
                        in
                            (gen_list b1 b2 0 0 d)
                        end
                  | h => [(SAME,h)]
                        )@(gen_list (t1 l1) (t1 l2) (p1+1) (p2+1) xs)
            else if p1 = x then
                (NEW,(hd l2))::(gen_list l1 (t1 l2) p1 (p2+1) d1)
            else
                (if new_only then
                    []
                 else
                    ((OLD,(hd l1)))::
                    (gen_list (t1 l1) l2 (p1+1) p2 d1))

fun elim_redund h_list =
    let
        (* combine around NEWERASE *)
        fun l_elim5 ((NEW,x)::(NEWERASE,y)::(NEW,z)::xs) =
            l_elim5 ((NEW,SEQ([x,y,z]))::xs)
          | l_elim5 (x::xs) = x::(l_elim5 xs)
          | l_elim5 [] = []

        (* get rid of NEWs around terms containing
         only MARKUPS *)
        fun l_elim4 ((NEW,x)::xs) =
            (if is_visible x then NEW else NEWERASE
                ,x)::(l_elim4 xs)
          | l_elim4 [x::xs] = (x::(l_elim4 xs))
          | l_elim4 [] = []

        (* collapse alternation of OLD/NEW/OLD/NEW *)
        fun l_elim3 ((OLD,o1)::(NEW,n1)::
                     (OLD,o2)::(NEW,n2)::xs) =
            l_elim3 ((OLD,SEQ([o1,o2]))::
                     (NEW,SEQ([n1,n2]))::xs)
          | l_elim3 [x::xs] = (x::(l_elim3 xs))
          | l_elim3 [] = []

        (* eliminate sequences of NEWs (OLDs) *)
        fun l_elim2 ((t1,x)::(t2,y)::xs) =
            if (t1 <> SAME2 andalso t1 = t2 then
```


WO 97/15890
PCT/US96/17142

```
                    l_elim2 ((i1,ISEQ([x,y]))::xs)
                else
                    ((i1,x)::(l_elim2 ((i2,y)::xs)))
              | l_elim2 (x::xs) = (x::(l_elim2 xs))
              | l_elim2 [] = []

            (* eliminate any NEW around something
             that is not visible *)
            fun l_elim21 ((NEW,x)::xs) =
                (if is_visible x then
                     (NEW,x)
                 else
                     (NEWERASE,x))::(l_elim21 xs)
              | l_elim21 (x::xs) = x::(l_elim21 xs)
              | l_elim21 [] = []

            (* eliminate any MARKUP that is OLD *)
            fun
                l_elim1 (l as ((OLD,RAW(s))::xs)) =
                (hd l)::(l_elim1 (tl l))
              | l_elim1 (l as ((OLD,(SEQ[i2]|BEGINNO(i2)))::xs)) =
                let
                    fun l_e (RAW(s)::xs) = (RAW(s))::(l_e xs)
                      | l_e (x::xs) = (l_e xs)
                      | l_e [] = []
                    val r = l_e i2
                in
                    if null r then
                        (l_elim1 (tl l))
                    else
                        (OLD,BEGINNO(r))::(l_elim1 (tl l))
                end
              | l_elim1 ((OLD,y)::xs) = (l_elim1 xs)
              | l_elim1 (x::xs) = (x::(l_elim1 xs))
              | l_elim1 [] = []

        in
            (l_elim5
             (l_elim4
              (l_elim3
               (l_elim2
                (l_elim21
                 (l_elim1(h_list))))))))
        end

    val (he1,_) = (eliminate_white (expl(h1)))
    val (he2,wh2) = (eliminate_white (expl(h2)))
    val s1 = (sentences (SEQ(he1)))
    val s2 = (sentences (SEQ(he2)))
    val DIFF(w,dif) =
        HtmlDiff.ilcs
        (Vector.vector s1) (Vector.vector s2)
in
    (merge_white
     (elim_redund (gen_list s1 s2 0 0 dif))
     (Vector vector wh2))
end
```

```
    fun do_si (SEQ([])) = (SEQ(map do_si []))
      | do_si m =
        if has_text m then
            SEQ([
                 MARKUP("STRONG",[]),
                 MARKUP("I",[]),
                 m,
                 ENDMARKUP("STRONG",[]),
                 ENDMARKUP("I",[])
                ])
        else
            m

    fun html_repl hp op_old op_new =
        let
            (* wrap around something new *)
            fun new_wrap h cnt top is_op =
                SEQ([
                     MARKUP("A",[("name","DIFF"^(makestring cnt))]),
                     ENDMARKUP("A",[]),
                     MARKUP("A",[("href","#DIFF"^
                                 (makestring
                                  (if cnt+1 = top then 0
                                   else cnt+1))
                                )]),
                     MARKUP("IMG",[("src",op_new)]),
                     ENDMARKUP("A",[])
                    ,
                    ((hmap.find(is_op,"A"))
                      handle hmap.NotFound => (SEQ([]))),
                    ,
                    [(do_si h)]
                    (case h of
                         (MARKUP(_,_)) => [h]
                       | (SEQ((MARKUP(_,_)::xs))) => [h]
                       | _ =>
                            [
                             MARKUP("STRONG",[]),
                             MARKUP("I",[]),
                             h,
                             ENDMARKUP("STRONG",[]),
                             ENDMARKUP("I",[])
                            ])
                    )

            fun old_wrap h cnt top is_op =
                SEQ([
                     MARKUP("A",[("name","DIFF"^(makestring cnt))]),
                     ENDMARKUP("A",[]),
                     MARKUP("A",[("href","#DIFF"^
                                 (makestring
                                  (if top = cnt+1 then 0
                                   else cnt+1))
                                )]),
                     MARKUP("IMG",[("src",op_old)]),
                     ENDMARKUP("A",[])
                    ,
                    ((((hmap.find(is_op,"A"))
                      handle hmap.NotFound => (SEQ([])))))
```


WO 97/15890
PCT/US96/17142

```
        0
        (
         MARKUP("STRIKE",[]),
         b,
       ENDMARKUP("STRIKE",[])
        ])

fun open_markup (SEQ([])|SEQ(NIL)) cc is_op =
    let fun iter ls cc is_op =
        if null ls then cc is_op
        else (open_markup (hd ls)
              (iter (tl ls) cc) is_op)
    in
        (iter t cc is_op)
    end
  | open_markup (m as MARKUP(s,_)) cc is_op =
    cc (hmap.insert(is_op,s,m))
  | open_markup (m as ENDMARKUP(s,_)) cc is_op =
    cc (((hmap.remove(is_op,s))
        handle hmap.NotFound => (is_op,RAW(""))))
  | open_markup h cc is_op = cc is_op

fun cnt_tag [((SAME|NEWERASE),_)::(t,s)::xs) cnt =
    if t <> SAME andalso t <> NEWERASE then
        cnt_tag xs (cnt+1)
    else
        cnt_tag ((t,s)::xs) cnt
  | cnt_tag ((t,_)::xs) cnt = (cnt_tag xs cnt)
  | cnt_tag [] cnt = cnt

val link_cnt = ref (cnt_tag hp 0)

fun is_same (((SAME|NEWERASE),_)::xs) = 1
  | is_same _ = 0

(* FIX: don't do anything until a body is found *)
fun repl ((NEW,(h))::xs) cnt is_op =
    (case hmap.peek(is_op,"HEAD") of
        NONE =>
            ((new_wrap h cnt (!link_cnt) is_op)
            ::
             (open_markup h (repl xs (cnt+(is_same xs)))
              is_op))
      | SOME _ =>
          (if ((is_same xs) = 1) then
               link_cnt := ((!link_cnt) -1
           else ());
           (h::
            (open_markup h (repl xs cnt) is_op))))
  | repl ((OLD,(h))::xs) cnt is_op =
    (case hmap.peek(is_op,"HEAD") of
        NONE =>
            ((old_wrap h cnt (!link_cnt) is_op)
            ::
             (open_markup h
              (repl xs (cnt+(is_same xs))) is_op))
      | SOME _ =>
                    ((if ((is_same xs) = 1) then
                        link_cnt := ((!link_cnt) - 1
                     else
                         ());
                     (SEQ([]))))::
                     (open_markup h
                      (repl xs cnt) is_op))
                    )
      | repl (((SAME|NEWERASE),(h))::xs) cnt is_op =
        (h::
         (open_markup h (repl xs cnt) is_op))
      (* close any open markups here *)
      | repl () _ _ = ())

    (* boilerplate for start of diff *)
    (fun click_here () =
        SEQ(
            [
             MARKUP("HR",[]),
             MARKUP("B",[]), RAW("#LnlDiff: "),
             ENDMARKUP("B",[]),
             RAW("Here is the "),
             MARKUP("A",[("href","#DIFF0")]),
             RAW("first difference."),
             ENDMARKUP("A",[]),
             RAW(" There are "^
                 (makestring (!link_cnt))^
                 " differences on this page."),
             MARKUP("IMG",[("src",op_old)]),
             RAW("<STRIKE>is old.</STRIKE>"),
             MARKUP("IMG",[("src",op_new)]),
             RAW("<STRONG><I> is new. </I></STRONG>"),
             MARKUP("HR",[])
             ])
        (fun body_pat (MARKUP("BODY",_)) = true
          | body_pat _ = false
        (fun body_act h =
            (unparse (SEQ([h,click_here()])))
    in
        (if ((!link_cnt) > 0) then
            unparse_pa body_pat body_act
            (SEQ(repl hp 0 (hmap.mkDict(1))))
        else
            unparse (SEQ(repl hp 0 (hmap.mkDict(1))))
    end


fun merge f1 f2 out only =
    let
        val f_out = open_out out
        val hl = (parsef f1)
        val m = html_merge (parsef f1) (parsef f2) only
        val res = (html_repl m
                   "http://www-spr.ih.att.com/~tball/pics/oold.gif"
                   "http://www-spr.ih.att.com/~tball/pics/onew.gif")
    in
        ((output f_out res); close_out f_out)
    end
```


WO 97/15890
PCT/US96/17142

```
                val j = ref (b_len-1)
                val M = ref 0
                val M_ind = ref b_len
            in
                (Alg0 a i b b_len L1 false );
                (Alg0 (a+i) (a_len-i) b b_len L2 true);
                M := (vt b_len);
                while !j >= 0 do
                    (
                     if (!M) <= (vt (!j)) then
                         (M := (vt (!j)); M_ind := (!j))
                     else
                         ();
                     j := !j - 1
                    );
                (
                 (AlgC (a+i) (a_len-i)
                  (b+(!M_ind)) (b_len-(!M_ind))
                  (AlgC a i b (!M_ind) result))
                )
            end

        fun cmb [] v l = DIFF(v,l)
          | cmb ((w,d)::xs) sum l = cmb xs (w+sum) (d::l)
    in
        (cmb (AlgC 1 m 1 n [])  0 [])
    end

fun diff (A : M.T vector) (B : M.T vector) =
    (p diff 0 0 A B)

(* Wu Yang's optimized LCS algorithm *)
fun ilcs (A : M.T vector) (B : M.T vector) =
    let
        open Array
        val weight = (M.weight diff)
        val m = Vector.length A
        val n = Vector.length B

        val Row0 = ref (array(m+1),0))
        val Row1 = ref (array(m+1),0))
        val Col0 = ref (array(n+1),0))
        val Col1 = ref (array(n+1),0))

        (* assumes that Row0 = row J I and Col0 = col I I
         * have already been computed *)
        fun find_pivot I J =
            let
                fun swap_ref (x,y) =
                    let val t = x in
                        (x:=!y;y:=(!t))
                    end
```

```
        fun fill st
            fill_fun esc_bool esc_fun end_fun
            let
                fun iter i =
                    if i < length(!ar) then
                        (update(!ar,i,fill_fun i);
                         if (esc_bool i) then
                             (esc_fun i)
                         else
                             iter (i+1))
                    else
                        (end_fun ())
            in
                iter st
            end

        fun simple_fill ar st fill_fun =
            fill ar st fill_fun
            (fn x => false) (fn x => ()) (fn x => ())

        fun comp_weight (r0,r1) (v0,v1) j i =
            let
                val DIFF(w,_) =
                    (weight
                     (Vector.sub(v0,i-1),
                      Vector.sub(v1,j-1)))
            in
                max(sub(!r0,i-1)+w,
                    max(sub(!r1,i-1),sub(!r0,i)))
            end

        fun verify_column (col,row) (x,y) =
            if col <= x then
                (fill Col1 row
                (comp_weight
                 (Col0,Col1) (ref B, ref A) col)
                (fn r => (sub(!Col1,r) > sub(!Row0,x)))
                (fn r =>
                 ((simple_fill Col1 (r+1)
                   (fn s => (sub(!Col1,r)))):
                  (swap_ref(Col0,Col1)):
                  (verify_row (col+1,y+1) (col,r-1)))
                (fn _ =>
                 (swap_ref(Col0,Col1);
                  verify_column (col+1,row) (x,y)))
            else
                (x,y)

        and verify_row (col,row) (x,y) =
            if row <= y then
                (fill Row1 col
                (comp_weight
                 (Row0,Row1) (ref A, ref B) row)
```


WO 97/15890
PCT/US96/17142

```
                    (fn c => (sub(Row1,c) > sub(Col0,y)))
                    (fn c =>
                     ((simple_fill Row1 (c+1)
                       (fn z => (sub(Row1,c))));
                      (swap_cell(Row0,Row1));
                      (verify_column (x+1,row+1) (c,row))))
                    (fn _ =>
                     (swap_ref(Row0,Row1);
                      verify_row (col,row+1) (x,y)))
                else
                    (x,y)

            fun find_non_zero (col,row):
                let val DIFF(w,_) =
                    weight(Vector.sub(A,col-1),
                           Vector.sub(B,row-1))
                in
                    if w <> 0 then
                        ((simple_fill Row0 z
                          (fn k => if k < col then 0 else w));
                         (verify_column (1,row+1) (col,row)))
                    else
                        if col < m then
                            (find_non_zero (col+1,row)
                        else if row < n then
                            (find_non_zero (1,row+1)
                        else
                            (m+1,n+1)
                end
        in
            ((simple_fill Row0 (1-1) (fn _ => 0));
             (simple_fill Col0 (J-1) (fn _ => 0));
             (find_non_zero (1,J)))
        end

    fun main i j (wt,res) =
        if i<=m andalso j<=n then
            let
                fun revres (x::xs) res = revres xs (x::res)
                  | revres [] res = res

                val (piv_x,piv_y) =
                    (if (Debug) then
                         (print "find_pivot ";
                          print i; print " "; print j; print "\n") else ();
                     (find_pivot i j))

                val DIFF(w,l) =
                    (if (Debug) then
                         (print "diff ";
                          print piv_x; print " ";
                          print piv_y; print "\n")
                     else ();
                     if (piv_x <= m) then
                         (p_diff (i-1) (j-1)
                         (Vector.tabulate(piv_x-i+1,
                                          (fn x =>
                                           Vector.sub(A,i+x-1))))
                         (Vector.tabulate(piv_y-j+1,
                                          (fn y =>
                                           Vector.sub(B,j+y-1)))))
                     else
                         DIFF(0,[]))
                in
                    (ident_prefix (piv_x+1) (piv_y+1)
                    (wt+w,(revres l res))
                end
            else
                DIFF(wt,(rev res))
        and ident_prefix x y (wt,res) =
            (if (Debug) then
                 (print "prefix "; print x;
                  print " "; print y;
                  print "\n") else ();
             if x <= m andalso y <= n then
                let val DIFF(w,l) =
                    weight(Vector.sub(A,x-1),
                           Vector.sub(B,y-1))
                in
                    if w <> 0 then
                        (ident_prefix (x+1) (y+1)
                         ((wt+w),(x-1,y-1,DIFF(w,l))
                          ::res))
                    else
                        main x y (wt,res)
                end
            else
                DIFF(wt,(rev res)))

        in
            main 1 1 (0,[])
        end
    end


structure CharW : WEIGHT =
    struct
        type T = char
        fun weight diff (a:T,b:T) =
            if a = b then DIFF(1,[]) else DIFF(0,[])
    end

structure CharDiff = Diff(STRUCTURE W = CharW );
```


WO 97/15890
PCT/US96/17142

```
signature HTML =
  sig
    type HtmlAtt

    datatype HtmlDoc =
        RAW of string
      | MARKUP of string * (HtmlAtt list)
      | ENDMARKUP of string * (HtmlAtt list)
      | DECLMARKUP of string * (HtmlAtt list)
      | PROCMARKUP of string * (HtmlAtt list)
      | BEGEND of (HtmlDoc list)
      | SEQ of (HtmlDoc list)

    (* get the names of input fields - TEXTAREA, INPUT, SELECT *)
    val info : string -> string option list * bool * bool * bool

    val info_stream : instream * string -> string option list * bool * bool * bool

    val parse : string -> HtmlDoc

    val parsef : string -> HtmlDoc

    val parse_stream : instream * string -> HtmlDoc

    val unparse : HtmlDoc -> string

    val unparse_pa : (HtmlDoc->bool) -> (HtmlDoc->string) -> HtmlDoc->string

    val domap : (HtmlDoc -> HtmlDoc) -> HtmlDoc -> HtmlDoc

    val getatt: string -> { string * string } list -> string option

    val hier : HtmlDoc -> HtmlDoc

    val expl : HtmlDoc -> HtmlDoc list

    val sentences : HtmlDoc -> HtmlDoc list

  end


structure Html : HTML =
    struct
      open StringUtil

      exception BadHtml

      type HtmlAtt = string * string

      datatype HtmlDoc =
          RAW of string
        | MARKUP of string * (HtmlAtt list)
        | ENDMARKUP of string * (HtmlAtt list)
        | DECLMARKUP of string * (HtmlAtt list)
        | PROCMARKUP of string * (HtmlAtt list)
        | BEGEND of (HtmlDoc list)
        | SEQ of (HtmlDoc list)

      val toupper = jcrlib.toupper

      fun maybe_str "" = ""
        | maybe_str s =
          (" "^(s,0); "\"" ^ expandStr s ^ "\" )
              handle NotFound => s

      fun unparse_attrs [(n,"")::t] = n ^ " " ^ unparse_attrs t
        | unparse_attrs [(n,v)::t] = n ^ "=" ^ (maybe_str v) ^ " " ^
                                     unparse_attrs t
        | unparse_attrs nil = ""

      fun unparse (RAW s) = s
        | unparse (SEQ l)|(BEGEND l)) = (foldr op ^ "" (map unparse l)
        | unparse (MARKUP(s,l)) = "<" ^ s ^ " " ^
                                  (unparse_attrs l) ^ ">"
        | unparse (ENDMARKUP(s,l)) = "</" ^ s ^ " " ^
                                  (unparse_attrs l) ^ ">"
        | unparse (DECLMARKUP(s,l)) = "<!" ^ s ^ " " ^
                                  (unparse_attrs l) ^ ">"
        | unparse (PROCMARKUP(s,l)) = "<?" ^ s ^ " " ^
                                  (unparse_attrs l) ^ ">"

      fun unparse_pa pat act (RAW s) =
              if pat (RAW s) then (act (RAW s)) else s
        | unparse_pa pat act (SEQ l) =
              if pat (SEQ l) then (act (SEQ l))
              else ((foldr op ^ "" (map (unparse_pa pat act) l))
        | unparse_pa pat act (BEGEND l) =
              if pat (BEGEND l) then (act (BEGEND l))
              else ((foldr op ^ "" (map (unparse_pa pat act) l))
        | unparse_pa pat act (MARKUP(s,l)) =
              if pat (MARKUP(s,l)) then (
                    (act (MARKUP(s,l)))
              ) else ("<" s "" ^^(unparse_attrs l)^^">")
        | unparse_pa pat act (ENDMARKUP(s,l)) =
              if pat (ENDMARKUP(s,l)) then (
                    (act (ENDMARKUP(s,l)))
              ) else ("</" s "" ^^(unparse_attrs l)^^">")
        | unparse_pa pat act (DECLMARKUP(s,l)) =
              if pat (DECLMARKUP(s,l)) then (
                    (act (DECLMARKUP(s,l)))
              ) else ("<!" s "" ^^(unparse_attrs l)^^">")
        | unparse_pa pat act (PROCMARKUP(s,l)) =
              if pat (PROCMARKUP(s,l)) then (
                    (act (PROCMARKUP(s,l)))
              ) else ("<?" s "" ^^(unparse_attrs l)^^">")


(* explode raws *)
fun expl (RAW s) =
    let
```


WO 97/15890
PCT/US96/17142

```
        open Format
        fun humpty (x::xs) word word_l white =
            let val w =
                (x = #" " orelse x = #"\t" orelse x= #"\n")
            in
                if white = w then
                    humpty xs (x::word) word_l w
                else
                    humpty xs [x]
                      (if null word then
                          word_l
                      else
                          (implode (rev word))::word_l)
                      w
            end
          | humpty [] word word_l white =
            if null word then
                word_l
            else
                (implode (rev word))::word_l)
        fun wrap_rev [] = []
          | wrap_rev (x::xs) = (RAW x)::(wrap_rev xs)(
    in
        wrap_rev(rev (humpty (explode s) [] [] true)))
    end
  | expl (SEQ l) = (foldr op @ [] (map expl l))
  | expl (BEGEND l) = (foldr op @ [] (map expl l))
  | expl a = [a]

fun sentences (doc) =
    let
        val no_breaks = ["B", "I", "TT", "U", "A", "IMG",
                         "DFN", "EM", "CITE", "CODE",
                         "KBD", "SAMP", "STRONG", "STRIKE",
                         "VAR", "FONT" ]
        fun comb ([],res) = res
          | comb (sent,res) = ([BEGEND(rev sent)]::res)
        fun hh ( x as [MARKUP(s,_)]|[ENDMARKUP(s,_)]::xs)
            (sent,res) =
            hh xs (if (List.exists (fn x=>x=s) no_breaks) then
                      (x::sent),res)
                  else
                      ([],(x::(comb (sent,res))))
          | hh ([SEQ l]::xs) (sent,res) = hh xs (hh l (sent,res))
          | hh ([RAW s]::xs) (sent,res) =
            let val e = hd(rev(explode s)) in
                if (e = #"." orelse e = #":" orelse e = #";"
                    orelse e = #"!" orelse e = #"?") then
                    hh xs ([].comb([RAW s]::sent,res))
                else
                    hh xs (([RAW s]::sent),res)
            end
          | hh (x::xs) (sent,res) = hh xs ((x::sent),res)
          | hh [] p = p
    in
        (case doc of
            SEQ sl => (rev (comb (hh sl) ([],[])))
          | x => [x])
    end

(* hierarchise markups with begin and end *)
fun hier (doc) =
    let
        fun add h ((m,l)::xs) = ((m,(h::l))::xs)
          | add h [] = [("",[h])]

        fun hh ((m as MARKUP(s,l))::xs) stk =
            if null stk orelse ((#1 (hd stk)) <> s) then
                hh xs (add m stk)
            else
                hh xs (add (BEGENDim::[]) (hd stk))) (tl stk))

          | hh ((m as ENDMARKUP(s,l))::xs) stk =
                hh xs ((s,[m])::stk)

          | hh ((SEQ l)::xs) stk =
                hh xs (add (hier (SEQ l)) stk)

          | hh (x::xs) stk =
                hh xs (add x stk)

          | hh [] ((m,l)::_) = l

          | hh [] [] = []
    in
        (case doc of
            SEQ sl => (SEQ (hh (rev sl) []))
          | x => hd (hh [doc] []))
    end

fun offer s =
    let val st = ref 0
    in fn i => if (!st) = 0 then (st := 1; s) else ""
    end

fun parse' next_tok =
    let
        open HtmlLex.UserDeclarations

        fun closeof (MARKUP(a,b)) = ENDMARKUP(a,b)
          | closeof _ = raise BadHtml

        fun procof (MARKUP(a,b)) = PROCMARKUP(a,b)
          | procof _ = raise BadHtml

        fun declof (MARKUP(a,b)) = DECLMARKUP(a,b)
          | declof _ = raise BadHtml

        fun norm acc =
            (case next_tok() of
```


WO 97/15890
PCT/US96/17142

```
    EOF => acc
  | CONTENT(s) => norm (RAW(s) :: acc)
  | (* "<" *) OPEN => norm (markup() :: acc)
  | (* "</" *)CLOSEOPEN =>
        let
            val mu = markup()
            val cl = closeof( mu )
        in
            norm (cl :: acc)
        end
  | (* "<!" *) DECLOPEN =>
        let
            val mu = markup()
            val cl = declof( mu )
        in
            norm (cl :: acc)
        end
  | (* "<?" *) PROCOPEN =>
        let
            val mu = markup()
            val cl = procof( mu )
        in
            norm (cl :: acc)
        end

  | x => norm acc)

and attlist first =
    (case first of
        ID(s) =>
            (case next_tok() of
                EQ => (case next_tok() of
                          ID(r) => (toupper s,r)
                        | STR(r) => (toupper s,r)
                        | _ => ("!BAD!","!BAD!")) :: (attlist(next_tok►

              | CLOSE => []
              | ID(r) => (toupper s,"") :: (attlist (ID(r)))
              | STR(r) => (toupper s,"") :: (attlist (STR(r)))
              | _ => raise BadHtml)
      | STR(s) => [("\"" ^ s ^ "\"" ,"") :: (attlist (next_tok()))
      | x => [])

and eat_comment s =
    (case next_tok() of
        CLOSECOMMENT => (s ^ "-- ")
      | COMMENT(more) => eat_comment (s ^ more)
      | _ => raise BadHtml)

and markup () =
    (case next_tok() of
        ID(s) => MARKUP(toupper s,attlist( next_tok() ))
      | OPENCOMMENT => MARKUP( "-- ",[(eat_comment "","")] )
      | x => MARKUP("",nil))
in
```

◄((((

```
        SEQ ( rev (norm []) )
    end

fun parse s =
    let
        val _ = HtmlLex.UserDeclarations.filename:=s
        val _ = HtmlLex.yylineno:= 0
        val lex = HtmlLex.makeLexer (offer s)
    in
        parse' lex
    end

fun parse_stream (in_f, f) =
    let
        val _ = HtmlLex.UserDeclarations.filename:=f
        val _ = HtmlLex.yylineno:= 0
        val lex = HtmlLex.makeLexer (inputc (in_f))
        val _ = LexHelp.SetFilename f
    in
        parse' lex
    end

fun parsef f =
    let
        val in_f = (open_in f)
        val lex = HtmlLex.makeLexer (inputc (in_f))
        val _ = HtmlLex.yylineno:= 0
        val _ = LexHelp.SetFilename f
        val res = parse' lex

    in
        ((close_in in_f); res)
    end


fun docmap f doc =
    (case doc of
        SEQ(l) => SEQ ( map (docmap f) l )
      | x => f x)

fun getatt aname ((n,v)::t) =
    if n = (jcrlib.toupper aname) then SOME v
    else getatt aname t
  | getatt aname [] = NONE

fun info_stream (inS, s) =
    let
            val html = parse_stream (inS, s)
            fun fields (MARKUP(s, attlist)) =
                    (case s of
                      "TEXTAREA" => [ getatt "NAME" attlist ]
                    | "SELECT" => [ getatt "NAME" attlist ]
                    | "INPUT" => [ getatt "NAME" attlist ]
                    | _ => [])
              | fields (SEQ(htmls)) =
```

```
                let
                        fun fields' (h::t) =
                                (fields' t)@(fields h)
                          | fields' [] = []
                in
                        fields' htmls
                end
            | fields _ = []
        fun submit (MARKUP(s, attlist)) =
                (case s of
                  "INPUT" => (
                        case getatt "TYPE" attlist of
                          SOME s => jcrlib.toupper s = "SUBMIT"
                        | NONE => false)
                ) _ => false)
          | submit (SEQ(htmls)) =
                let
                        fun submit' (h::t) =
                                if submit h then true
                                else submit' t
                          | submit' [] = false
                in
                        submit' htmls
                end
          | submit _ = false
        fun has_markup desired (MARKUP(s, _)) =
                (jcrlib.toupper s) = desired)
          | has_markup desired (SEQ(htmls)) =
                let
                        fun has_markup' (h::t) =
                                if has_markup desired h then true
                                else has_markup' t
                          | has_markup' [] = false
                in
                        has_markup' htmls
                end
          | has_markup _ _ = false
    in
        (fields html, submit html, has_markup "BODY" html, has_markup "/BODY"
" html)
    end

fun info s =
    let
        val inS = open_in s
    in
        info_stream (inS, s) before close_in inS
    end

end
```


WO 97/15890
PCT/US96/17142

```
datatype lexresult =
        OPEN |
        ENDOPEN |
        DECLOPEN |
        PROCOPEN |
        CLOSE |
        OPENCOMMENT |
        CLOSECOMMENT |
        EQ |
        CONTENT of string |
        COMMENT of string |
        STR of string |
        ID of string |
        EOF

val eof = fn () => EOF
val filename = ref ""
(*
fun dprint(s) = (print (s ^ "(" ^(makestring (0)) ^ ")\n"))
*)
fun dprint(s) = ()

val error = fn (x,l:int) => output(std_err,x ^ " line " ^
                (makestring (1+l)) ^
                ", file `" ^ (!filename) ^ "'\n")

%%
%s MU COMM COMMSTUP;
%structure HtmlLex
ws      = ([\t\ \n])+;
nonws   = ([^\ \t\n<>=])+;
%count
%%
<INITIAL>[^<>]* => (dprint "CONTENT"; CONTENT(yytext));
<INITIAL>"<" => (YYBEGIN MU; dprint "OPEN"; OPEN);
<INITIAL>"</" => (YYBEGIN MU; dprint "ENDOPEN"; ENDOPEN);
<INITIAL>"<!" => (YYBEGIN MU; dprint "DECLOPEN"; DECLOPEN);
<INITIAL>"<?" => (YYBEGIN MU; dprint "PROCOPEN"; PROCOPEN);
<MU>">" => (YYBEGIN INITIAL; dprint "CLOSE"; CLOSE);
<MU>"--" => (YYBEGIN COMM; dprint "OPENCOMMENT"; OPENCOMMENT);
<MU>"=" => (dprint "EQ"; EQ);
<MU>\"[^\n"]*\" => (dprint "STR"; STR(substring(yytext, 1, String.size(yytext)-2)));
<MU>{nonws} => (dprint "ID"; ID(yytext));
<MU>{ws} => (lex());
<COMM>"--" => (YYBEGIN MU; dprint "CLOSECOMMENT"; CLOSECOMMENT);
<COMM>"<" => (YYBEGIN INITIAL; dprint "CLOSECOMMENT";
(yybufpos := (!yybufpos)-1); CLOSECOMMENT);
<COMM>[^-<>]* => (dprint "COMMENTTEXT"; COMMENT(yytext));
.       => (error("html lex:  ignoring bad character "^yytext, !yylineno); lex());
%%
```

```
#!/usr/local/bin/perl -- # A comment mentioning perl, to prevent looping

eval 'exec perl -S $0 ${1+"$@"}'
    if 0;

# For the latest info on Fred Douglis's update of v1.0 of w3newer, see
#
# http://www.ll26.research.att.com/~douglis/track_urls/
#
# For the latest information on Brooks Cutter's version of w3new, check the URL
#
# http://www.eng.paradyne.com/cgi-bin/bbcurn?user=scutter&page=w3new
#
# w3newer v1.1 - Creates a What's New list of http: URL's
#
# Modified by Fred Douglis <douglis@research.att.com>
# to separate user interface from polling.
#
# w3newer is a program that will extract a list of URL's from either your Mosaic
# hotlist, or will extract the URL's from a HTML document.  It will then obtain
# the HTTP/1.0 modification dates for each document listed, and output a HTML
# file with the URL's sorted by their last modification time.
#
#
# Package: w3newer
# Author: Brooks Cutter (bcutter@paradyne.com)
# Author: Fred Douglis <douglis@research.att.com>
# Latest version: 1.0
# Last updated: March, 1995
# Archive: w3newer.tar.gz
# (includes everything you need to run it, except perl v4.036)
#
#
# What it does.
#
#  1. w3newer first extracts the URL's from your hotlist or a HTML document, and
#     cached modification dates and access times from its own cached
#     history and the WWW browser's history file.
#  2. w3newer then hands off the URLs to a separate process, which
#     proceeds to do a HTTP/1.0 HEAD on each http: URL that might
#     be out of date and stores the Last modified time of each URL (if
#     available).
#  3. Finally, w3newer sorts its output based on the last modified time of the
#     document and categorizes them by the month they were modified.  For
#     non-http URL's or URL's for which it can't retrieve the Last modified
#     time, it will list these at the bottom of its HTML output.
#
# This program was written because Brooks found himself frequently checking web
# pages in his hotlist to see if they had recently been changed.  Now
# he runs w3newer from cron nightly, and checks its output each morning.
#
# pointing w3newer at a list of URL's
# ***********************************
#
#  1. Using your Mosaic hotlist by default.  w3newer will read your Mosaic
#     hotlist file and extract the document URL's and document titles.  It will
#     look for your hotlist in your home directory as the file
#     ~/.mosaic-hotlist-default
#
#     this can be overriden by using the environment variable
#     W3NEWER_HOTLIST
#
#     The default and environment variable can be overridan with the
#     command line arguments -hotlist_fn or -l with a argument of the file in
#     Mosaic-hotlist format.
#  2  Extracting URL's from a HTML document if you call w3newer with the
#     -u argument and a HTML document URL.  It will retrieve that document,
#     extract all the links in the document, and check the modification time of
#     each link.
#  3  Extracting URL's from a HTML document and inline modification times
#     if you call w3newer with the -html argument and a HTML document
#     URL.  It will extract the document links and retrieve the modification
#     times, but unlike #2 (-u argument), it will replace the modification
#     times within the document rather than producing a sorted list.
#
#     When w3newer parses the document, it will look for a tag of the form
#
#     <w3newer url="URL">
#     For example
#     <w3newer url="http://www.ncsa.com/file.html">
#
#     When it finds a tag like the one above, it will retrieve the last
```

```
    *    modification time for the document specified by the url= line. and
    *    include the last modification of that URL (if it exists)
    *
    *
    * For more information on the usage, run w3newer with the -? argument
    *
    * The latest information on this program can be found on the web - see above
    *
    * This program was built with software written by others.
    * Their contributions are greatly appreciated..
    *
    * libwww-perl v0.40 or later by Roy Fielding (fielding@ics.uci.edu)
    *    http://www.ics.uci.edu/WebSoft/libwww-perl/
    * evaluate_parameters (evap) by Stephen C. Licie (lusol@Lehigh.EDU)
    *    All the implementations of evaluate_parameters are available via
    *    anonymous FTP from ftp.Lehigh.EDU (128.180.6) 4). Look in the
    *    pub/evap/evap-2.x directory for the latest compressed tar file.*
    * perl v4 036 by Larry Wall (lwall@netlabs.com)
    *    ftp able from ftp.uu.net in /systems/gnu as perl-4.036.tar.gz
    * w3new v0.4 by Brooks Cutter (bcutter@paradyne.com)
    *
    *
    * Known Bugs
    * **********
    *
    *  o none (that I'm aware of....
    *
    * Future work
    * ***********
    *
    *  o none (probably). I started to write some code that would do a checksum
    *    on a document specified by a URL ... this would allow you to track the
    *    modification time based on when the content changed... however I
    *    never finished coding/testing the checksum piece, and abandoned it.
    *    you'll find the code commented out below..
    *  - picked up 1/95 by F. Douglis.  Cached modification times are positive;
    * cached checksums are negative.
    *  o eventually I hope to convert this into a WWW applet and let the
    *    browser do the work. You'll find more info on WWW Applets at
    *    http://www.let.rug.nl/~bert/WIA/WIA.html
    *
    *  o Other future work (FD): multi-user 'database' of info.  Types other than
    *    http.
    *
    * Brooks Cutter (bcutter@paradyne.com)
    * http://www.eng.paradyne.com/cgi-bin/bbcurn?user=bcutter
    *-----------------------------------------------------------------------
    * w3newer is distributed under the Artistic License (included with your Perl
    * distribution files and with the distribution of the libwww-perl package).
    *-----------------------------------------------------------------------

    * To run this program either set the variable below with the location
    * $ENV{'W3NEWER'} = '/home/tball/src/w3newer' unless ($ENV{'W3NEWER'});
    $ENV{'W3NEWER'} = '/home/tball/src/w3newer';
    $w3newer_dir = $ENV{'W3NEWER'};
    unless($w3newer_dir) {
      print <<EOF;

    ***ERROR***  Unable to start w3newer

    Sorry, I don't know where w3newer support files are installed!

    Please either set the variable \$w3newer_dir near the top of the w3newer script,
    or set the W3NEWER environment variable to the base package directory.

    EOF
      exit 1;


    !process_urls = 'perl $w3newer_dir/lib/w3newer-get.pl';

    ...ess($ENV{'http_proxy'}) {
        $ENV{'no_proxy'}='www1126.research.att.com,spo.att.com,radish.research.att.com,localhost.tbu.att.com,ho.att.com';
        $ENV{'http_proxy'}='http://www.ih.att.com:8050/';


    * What version are you using? (set this if using $w3newer_dir)
    $lib_www_ver = '5.40';
    $ENV{'LIBWWW_PERL'} = '$w3newer_dir/lib/libwww-per.-$lib_www_ver';
        if (defined $ENV{'LIBWWW_PERL'});
    unshift(@INC,
      '$w3newer_dir'
```

```
    '$w3newer_dir/lib',
    # whereis libwww-perl?
    # Either in the directory specified by LIBWWW_PERL
    # or under lib in $w3newer_dir
    $ENV{'LIBWWW_PERL'},
    '/home/douglis/lib/perl',
);

$whats_new = 'perl $ENV{'LIBWWW_PERL'}/POST http://hotwab.tbu.att.com/~douglis/cgi-bin/whatsnew.cgi';

# architecture-specific library (for sys/socket.ph:
require 'arch.pl';
$arch = &arch;
if (-d ($libloc = '$w3newer_dir/lib/arch/$arch')) {
    unshift(@INC, $libloc);
}

# TBALL: Let's keep some usage stats
open(MAIL,'| mailq tball');
$date = 'date';
print MAIL 'Subject: w3newer\n$ENV{'USER'}\n$date\n';
close(MAIL)
# TBALL

require 'ctime.pl';
#require 'request.pl';  # part of www nov
require 'stat.pl';
require 'www.pl';
require 'wwurl.pl';
require 'wwerror.pl';
require 'wwdates.pl';
require 'wwmot.pl'; # part of libwww-perl v0.20 + 1 later..

unshift(@INC,'/appl/evap/lib') if (-d '/appl/evap');
require 'evap.pl';

require 'parse_html.pl';

# You shouldn't need to change anything below here..
$gl_request_timeout = 45;

@months = (
    'January','February','March','April','May','June','July',
    'August', 'September','October','November','December'
);
@days=('Sunday','Monday','Tuesday','Wednesday','Thursday','Friday', 'Saturday');

$w3newer_author_home = 'stuff.com';
$w3newer_ver = '1.0';
$w3newer_author_email = 'bcutter@$w3newer_author_home';
$w3newer_author = <<EOF;
<a href="http://www.$w3newer_author_home/cgi-bin/bbcurn?user=bcutter">
Brooks Cutter</a>
EOF
$w3newer_oldurl =
    'http://www.stuff.com/~bcutter/home/programs/w3new/w3new.html';
$w3newer_url =
  'http://www1126.research.att.com/~douglis/track_urls/';

$w3newer_author2 = <<EOF;
<A HREF="http://www.research.att.com/orgs/ssr/people/douglis/">Fred Douglis</A>
EOF

# if URL isn't fully qualified, this is used with &wwurl'absolute()
$base = 'file://localhost'. ($ENV{'PWD'} || $ENV{'cwd'}) .'/';

  $UserAgent = 'w3newer/$w3newer_ver $www'Library'; # Set up User-Agent: header
# Set the default User-Agent
  &www'set_def_header('http','User-Agent',$UserAgent); #'

  @hotlist_netscape = ("$ENV{'HOME'}/.netscape-bookmarks.html",
                       "$ENV{'HOME'}/.MCOM-bookmarks.html");
  @history_netscape = ("$ENV{'HOME'}/.netscape-history",
                       "$ENV{'HOME'}/.MCOM-global-history");
  @hotlist_mosaic = "$ENV{'HOME'}/.mosaic-hotlist-default";
  @history_mosaic = "$ENV{'HOME'}/.mosaic-global-history";

  $default_browser = 'netscape';

# This requires the evap.pl library..
$evap_pc=<<EOF;
??? w3newer
```

```
# the following options control output processing
  verbose, v: switch
  output_fn, o: file = W3NEWER_HTML, '~/public_html/w3newer_output.html'
# If you prefer to have it output to STDOUT, uncomment the next line..
#  output_fn, o: file = W3NEWER_HTML, stdout
# the following control where to get URLs
  mosaic, ms: switch
  netscape, ns: switch
  hotlist_fn, l: file = W3NEWER_HOTLIST, ''
  history_fn, h: file = W3NEWER_HISTORY, ''
  url, u: string
  html_url, hurl, string
# these control processing
  from_address, from: string = EMAIL_ADDRESS, ''
  checksum, cs, boolean = TRUE
# these control the format of the output
  display_header, dh: boolean = TRUE
  display_footer, df: boolean = TRUE
  display_unknown, dU: boolean = TRUE
  display_recent, dr: boolean = TRUE
  display_unchanged, du: boolean = TRUE
  embolden, b: switch
  last_visited, lv: switch
  timestamp_source, ts: switch
# these control cached information
  cache_modtimes, cm: boolean = TRUE
  cm_file, cmf: file = W3NEWER_CMFILE, '~/.w3newer_modtimes'
  cm_def_threshold, cmdt: string = W3NEWER_CM_DEF_THRESH '1d'
  want_latest, wl: boolean = FALSE
  cm_obsolete_threshold, cmot: string = W3NEWER_CM_OBS_THRESH '3d'
  cache_thresholds_fn, ctfn: file = W3NE  R_CTFILE '~/.w3newer_thresholds'
# miscellaneous options, esp. for debugging
  debug, d: integer = 0
  max_urls, max_urls: integer = -1
  multi_user, mu: switch
#  nop, n: switch = false
# integrated into modtimes
#  w3newer_cs_file, csf: file = W3NEWER_CSFILE, '~/.w3newer_checksums'
  snapshot, s: switch
  snapshot_url, su: string = W3NEWER_SNAP_URL, 'http://radish.research.att.com/cgi-bin/no_hands'
  ignore_noncs, ign: switch
  forms, forms: switch
POSTEND optional_file_list
EOF
$evap_pdt = split(/\n/,$evap_pdt);

$evap_ep <<EOF;
w3newer v$w3newer_ver

  w3newer is a program that will extract a list of URL's from a hotlist
  (either Netscape or Mosaic format), or will extract the URL's from an
  HTML document.  It checks the modification dates of URLs that have not
  been checked recently and produces a report of URLs that have been modified
  since they were viewed.

  It tries not to poll things too often.  It uses the following techniques:
    * If your browser's history shows something has been accessed recently, it
      isn't checked.
    * If w3newer checked something recently, it isn't checked.
    * If w3newer knows something has changed since it was last visited,
      it marks it as new but doesn't check for a more current date
      unless it deems the modification date information to be
      'obsolete' (see documentation for cm_obsolete_threshold).
    * The definition of 'recent' can vary from URL to URL, based on
      specifications in a 'thresholds' file.

  Currently, w3newer only handles http URLs, not ftp, file, or others.
  This may change

  Examples

    w3newer
    w3newer -usage_help
    w3newer -?
    w3newer -full_help
    w3newer -ns -b -lv -v    (netscape, bold, last visited, verbose)
verbose
  When verbose is set, w3newer will print out the name of each URL
  it checks and generate some other status information
checksum
  When checksum is set, w3newer will use checksums to tell if something
  has been modified if the modification date is unavailable.
```

.output_fn
This specifies where the HTML output is placed (stdout indicates standard output).
.display_header
determines whether or not to print a header on the HTML output
.display_footer
determines whether or not to print a footer on the HTML output
.display_unknown
determines whether or not to display hosts for which it cannot find HTTP/1.0 Last modification dates
.display_recent
determines whether or not to display URLs that were recently accessed and not checked during this run.
.display_unchanged
determines whether or not to display URLs that have not changed since they were last seen.
.hotlist_fn
The filename of your Netscape/Mosaic hotlist. By default it is set to the file ~/.netscape-bookmarks.html or ~/.mosaic-hotlist-default. depending on whether the 'netscape' or 'mosaic' option is set. The default can be overridden with the environment variable W3NEWER_HOTLIST.
.history_fn
The filename of your Netscape/Mosaic history. By default it is set to the file ~/.netscape-global-history or ~/.mosaic-global-history. depending on whether the 'netscape' or 'mosaic' option is set. The default can be overridden with the environment variable W3NEWER_HISTORY
.url
w3new will retrieve the HTML file specified by this argument, extract all hyperlinks and images from the document, and proceed to check the last modification time for each URL.
.html_url
If you call w3newer with the -f tml argument and a HTML document URL, it will extract the docu..t links and retrieve the modification times, but unlike the previous option (-u argument), it will rep..ace the modification times within the document rather than producing a sorted list.

When w3newer parses the document, it will look for a tag of the form

<w3newer url="URL">
For example
<w3newer url="http://www.host.dom/file.html">

When it finds a tag like the one above, it will retrieve the last modification time for the document specified by the url= line, and include the last modification of that URL (if it exists!
.from_address
By default, the program will set the HTTP/1.0 From: header to your Username (specified by USER environment variable) and your hostname. If your hostname isn't fully qualified (ie: doesn't have a '.' in it) then it will run 'domainname' and tack on the result (if any).

This value is also used for the snapshot arguments (see below).

This value can be overriden by setting the environment variable EMAIL_ADDRESS or passing your email address with the -from flag
.multi_user
This flag is not yet implemented. It refers to sharing modification dates among multiple users.
.netscape
This flag indicates that Netscape bookmark and history files should be used.
.mosaic
This flag indicates that Mosaic hotlist and history files should be used.
.emphlev
This flag indicates that certain items that are deemed to be of special interest should be generated in boldface. Currently this refers to the dates of items that have been modified since they were last seen
.last_visited
This flag indicates that the time when a URL was last visited, according to th history file, should be included in w3newer output.
.timestamp_source
This flag indicates that the source of information for a URL (proxy-caching server, polling, previous run, etc.) should be included in w3newer output.
.cache_fn
This option specifies the file that caches modification dates of URLs. The default can be overridden by the environment variable

```
#!NEWER_CONFILE.
.cache_modtimes
  This option indicates that modification dates should be cached.  If
  FALSE, cm_file is ignored.
.cm_def_threshold
  This option specifies the default frequency for checking URLs, in
  the form Nm, Nh, Nd, Nw for N minutes, hours, days, or weeks, respectively
.want_latest
  If this switch is set, always get the current modification date of a
  URL that has not been accessed recently, rather than skipping it if
  its known modification date is more recent than the time it was
  visited by the browser and not beyond the date specified by the
  -cmdt switch
.cm_obsolete_threshold
  This switch specifies when a cached modification date should not be
  considered to be current even if it indicates that a URL has not
  been seen since it was modified.
.cache_thresholds_fn
  This option specifies the file that specifies how often different
  URLs should be checked.  If it does not exist, then all URLs are
  checked with a frequency determined by the -cmdt option.  The format
  of this file is:
       URL <whitespace> time_specification
       Default          time_specification
       Obsolete         time_specification
  Default overrides the value of -cmdt.  Obsolete overrides -cmot.  If
  time_specification is 'never' the URL is skipped even if it is found
  in the hotlist.  The URL can be a pattern to match against (perl regexp
  syntax).
.debug
  When debug is > 0.  Additional debugging output is generated.  The higher
  the number, the more output there is.
.max_urls
  This option specifies a limit on the number of URLs to process.
  primarily for debugging purposes.
.snapshot
  If set, the output will contain links to the snapshot CGI daemon.
.snapshot_url
  URL to use for snapshot.
.ignore_nonets
  Set in order to bypass saved information about which URLs disallow robots (all or none).
.forms
  Set to produce a form for each URL rather than direct hyperlinks.
EOF
@evap_em = split(/\n/,$evap_em);

die "Error returned from evaplib\n" if (&evap('evap_pdt', *evap_em) != 1); #
# If from_address was set, set the default header..
if ($options{'from_address'}) {
    &www'set_def_header('http','From',$options{'from_address'}); #
} else {
    local($user) = $ENV{'USER'} || getlogin || (getpwuid($<))[0] || 'Intruder!!';
#   local($host) = $ENV{'HOST'} || chop($foo = `/usr/ucb/hostname`) || 'Unknown_host';
    local($host) = $hostname'FQDN;
    $options{'from_address'} = join('@', $user, $hostname'FQDN);
}

$option_string = "";
for ('Diff', 'Remember', 'History') {
    $option_string .= "<option> $_ ";
}

if (!$options{'output_fn'} eq '>-') || !$options{'output_fn'} =~ /^[a-z]/)) {
  open(OUT,$options{'output_fn'})
  || die "Can't write to $options{'output_fn'}: $!\n";
} else {
    local($oldmask) = umask;
    if (-e $options{'output_fn'}) {
        local(@sry) = stat(_);
        local ($mode) = $sry[$ST_MODE];
        umask($oldmask | (~$mode & 0777));
        if (-s $options{'output_fn'}) {
            rename($options{'output_fn'}, "$options{'output_fn'}~") ||
                print STDERR "Warning: can't rename $options{'output_fn'}\n";
        }
    }

  open(OUT,">$options{'output_fn'}")
      || die "Can't write to $options{'output_fn'}: $!\n";
    umask $oldmask;


$options{'verbose'} = 1 if $options{'debug'};  # debug implies verbose.
```

```
if (!$options{'mosaic'} || $options{'netscape'} ||
    $options{'hotlist_fn'} || $options{'url'} || $options{'html_url'})) {
    $options{$default_browser} = 'TRUE';
}

sub newcmurl {
    local ($url) = @_;
    if ($options{'debug'} > 0) {
        printf STDERR "Got a URL (%s) we didn't ask for -- We know about:",
        local ($u);
        for $u (sort keys %urls) {
            print(STDERR "\t%s\n", $u);
        }
    }
}


sub find_first_existing {
    local (@names) = @_;
    local($file);
    foreach $file (@names) {
        if (-e $file) {
            return $file;
        }
    }
    printf("Can't find an existing file from:\n %s\n", join("\n  ", @names)
        if $options{'debug'};
    return "";
}

if ($options{'mosaic'}) {
    $options{'hotlist_fn'} = &find_first_existing(@hotlist_mosaic);
    $options{'history_fn'} = &find_first_existing(@history_mosaic);
} elsif ($options{'netscape'}) {
    $options{'hotlist_fn'} = &find_first_existing(@hotlist_netscape);
    $options{'history_fn'} = &find_first_existing(@history_netscape);
}

&load_checksum if ($options{'checksum'});


# read in cached modification dates & checksums, if any. Then if we
# see URLs with cached info, update the cached info based on any new
# info, e.g. 'last seen' dates.

if ($options{'cache_modtimes'}) {
    %cached_modtimes = &read_cached_modtimes($options{'cm_file'});
} else {
    %cached_modtimes = ();
}

if ($options{'history_fn'} ne "") {
    %visit_times = &load_history($options{'history_fn'});
}

if ($options{'url'}) {
  $url = &wwwurl'absolute($base,$options{'url'});
  if (!&wwwbot'allowed($url,$UserAgent)) {
    die
    "Sorry, but the remote site does not allow robots to retrieve that URL.\n";
  }
  %urls = &extract_links_desc($url);
} elsif ($options{'html_ur'}) {
    $url = &wwwurl'absolute($base,$options{'html_url'});
    if (!&wwwbot'allowed($url,$UserAgent)) {
        die
        "Sorry, but the remote site does not allow robots to retrieve that URL.\n";
    }
print OUT &display_html($url);
exit;
} else {
    # use 'hotlist_*' names to have qualified pattern matching on call by ref--
    # is this necessary?
    local(%hotlist_urls,%hotlist_dates);
&load_hotlist($options{'hotlist_fn'}, *hotlist_urls, *hotlist_dates);
%urls = %hotlist_urls;
%urls_accessed = %hotlist_dates; #
for $url (keys %urls) {
    if ($urls_accessed{$url}  > $visit_times{$url}) {
        print STDERR "Warning: Visit time (from hotlist for $url newer than global history !\n" if $options{'debug'} && $visit_times{$url} > 0.
        $visit_times{$url} = $urls_accessed{$url};
```

```
        $stamp{$url} = "hotlist";
    }
  }

  if (length keys %visit_times == 0) {
      delete $options{'embolden'};
  }

sub threshold {
    local ($url) = @_;
    return $threshold{$url} if defined($threshold{$url});
    local($u);
    foreach $u (@thresholds) {
        if ($url =~ m"$u") {
            printf STDERR "Using threshold for %s for URL %s\n", $u, $url if $options{'debug'} > 0;
            return $threshold{$u};

    return $options{'cn_def_threshold'};

# initialize units (in terms of seconds)
#
    local(@units) = ('m', 60, 'h', 60, 'd', 24, 'w', 7);
    local($current) = 1;
    local($thisunit, $thismult);
    while (@units > 1) {
        ($thisunit, $thismult, @units) = @units;
        $units{$thisunit} = $thismult * $current;
        $current = $units{$thisunit};
    }
}

sub str2secs {
    local ($orig_thresh) = @_;
    local ($m,$num,$unit,$t) = 0;
    $t = $orig_thresh;
    while ($t ne "") {
        if ($t =~ /never/) {
            return -1;
        } elsif ($t =~ /^(\d+)([mhdw])(.*)/) {
            $num = $1, $unit = $2, $t = $3;
            $m += $num * $units{"\l$unit"};
        } elsif ($t =~ /^\d+$/) {
            $m += $t;
            return $m;
        } else {
            die "Can't parse threshold $orig_thresh"
        }
    }
    return $m;
}

sub secs2str {
    local ($s) = @_;
    local ($u,$res,$n);
    if ($s < 0) {
        warn "Negative time ($s) passed to routine";
        return "";
    }
    $res = "";
    for $u ('w', 'd', 'h', 'm') {
        die "Units not initialized" if !defined($units{$u});
        $n = 0;
        while ($s >= $units{$u}) {
            $n++;
            $s -= $units{$u};
        }
        $res .= "$n$u" if $n;
    }
    $res .= "${s}s" if $s;
    $res = "-" if $res eq "";
    $res;
}


if (-e $options{'cache_thresholds_fn'}) {
    local($set_default,$set_complete) = 0;
    open(THRESH, $options{'cache_thresholds_fn'}) || die "Can't open $options{'cache_thresholds_fn'}";
    printf(STDERR "Thresholds:\n") if $options{'debug'} > 1;
    while (<THRESH>) {
        if (/^Default $/) {
```

```
        $options{'cm_def_threshold'} = &str2secs($1);
        $set_default = 1;
        printf(STDERR "\t%-30s: %d\n", "Default", $options{'cm_def_threshold'}) if $options{'debug'} > 2;
    } elsif (/^Obsolete\s+(.*)/i) {
        $options{'cm_obsolete_threshold'} = &str2secs($1);
        $set_obsolete = 1;
        printf(STDERR "\t%-30s: %d\n", "Obsolete", $options{'cm_obsolete_threshold'}) if $options{'debug'} > 2;
    } elsif (/^\#/) {
        next                    # skip comment
    } else {
        local($url, $main) = split(/\s+/);
        if ($main ne "") {
            $threshold{$url} = &str2secs($main);
            push(@thresholds, $url);

        printf(STDERR "\t%-30s %d\n", $url, $threshold{$url}) if $options{'debug' > 2};

}
close(THRESH);
if (!$set_default
    $options{'cm_def_threshold'} = &str2secs($options{'cm_def_threshold'});

if (!$set_obsolete)
    $options{'cm_obsolete_threshold'} =
        &str2secs($options{'cm_obsolete_threshold'});




$urls_processed = 0;
@new_urls = @old_urls = @skipped_urls = @new_chksums = @old_chksums =
    @inaccessible_urls = @nobots = ();

print STDERR "Current time is ", time, "\n" if $options{'debug'}.

# create pipes to talk to the server -- this should be changed to use POST
# directly.
pipe(GETURLS, SENDURLS) || die "Can't create pipe: $!";
pipe(GETRESULTS, SENDRESULTS) || die "Can't create pipe: $!";
if ($pid = fork) {
    close(GETURLS).
    close(SENDRESULTS);
} elsif (defined $pid) {
    close(SENDURLS,
    close(GETRESULTS);
    open(STDOUT, ">&SENDRESULTS") || die "Can't dup stdout";
    open(STDIN, "<&GETURLS") || die "Can't dup stdin";
    exec $w3ata_new || die "Can't exec POST: $!";
} else {
    die "Can't fork: $!";
}

$i = 0.

for $url (keys %urls) {
    local($t) = &threshold($url);
    next if ($t == -1);
    printf STDERR "THR_SRV: $url threshold $t\n" if ($options{'verbose'}).
    next unless($url =~ /^http:/);
    next if ($url =~ m!^http:.*\?!);     # skip scripts
    printf(SENDURLS "%sURL%d=%s\n", ($i > 0) ? "&" : "", $i, $url).
    $i++.
}
close(SENDURLS)
while(<GETRESULTS>) {
    print(STDERR "<---$_") if $options{'debug'} > 1.
    next if    ^http:/.
    chop.
    ($url, $mod, $when, $rest) = split(/\s+/);
    die "Bad input from back end" if $rest ne "";
    &nosuchurl($url), next if !defined $urls{$url};
    next if $mod eq "N/A";
    local($cached_mod, $cached_checked, $nobots) =
        split(/:/, $cached_modtimes{$url});
    $nobots = "" if $options{'ignore_nobots'};
    if (($mod > $cached_mod) && ($cached_mod != 0) ||
        ($mod == $cached_mod && $when > $cached_checked)) {
        print STDERR "SRV: $url mtime $mod when $when\n" if $options{'verbose'}.
        $when = $cache_checked if $cache_checked > $when.
```

```
        $cached_modtimes{$url} = join($;, $mod, $when, $nobots);
        $stamp{$url} = 'proxy-caching server';
    } elsif ($mod > 0) {
        print(STDERR "old info for $url\n") if $options{'debug'} > 0;
    }
}


# create pipes to talk to the local back end.
pipe(GETURLS, SENDURLS) || die "Can't create pipe: $!";
pipe(GETRESULTS, SENDRESULTS) || die "Can't create pipe: $!";
if ($pid = fork) {
    close(GETURLS);
    close(SENDRESULTS);
} elsif (defined $pid) {
    close(SENDURLS);
    close(GETRESULTS);
    open(STDOUT, ">&SENDRESULTS") || die "Can't dup stdout";
    open(STDIN, "<&GETURLS") || die "Can't dup stdin";
    exec $process_urls || die "Can't exec $process_urls, $!";
} else {
    die "Can't fork: $!";
}

for $url (keys %urls) {
    local($t) = &threshold($url);
    if ($t == -1) {
        print STDERR "Skipping ignored URL $url\n" if $options{'debug'};
        next;
    }
    print STDERR "CHK: $url threshold $t\n" if $options{'verbose'};
    next unless $url =~ /^http:/ || $url =~ /^file:/;
    next if $url =~ m!^http:.*\?!; # skip scripts
    last if $options{max_urls} >= 0 && --$urls_processed > $options{max_urls};

    $last_checked = $mod = $visit_times{$url}; $cached_cs = 0;
    if ($url =~ /^http:/) {
        if (defined $cached_modtimes{$url}) {
            ($cached_mod, $cached_checked, $nobots) =
                split($;, $cached_modtimes{$url});
            if ($cached_checked >= $last_checked) {
                $last_checked = $cached_checked;
                $stamp{$url} = 'previous run';
            }
            if ($cached_mod > 0) {
                if ($cached_mod > $visit_times{$url} &&
                    (time - $cached_mod <=
                     $options{'tm_obsolete_threshold'}) &&
                    !$options{'want_latest'}) {
                    push(@new_urls, $url);
                    $urls_time{$url} = $cached_mod;
                    print STDERR "$url known to be new: skipping\n" if $options{'verbose'};
                    if ($options{'debug'}) {
                        local($mdate) = &ctime($urls_time{$url});
                        local($vdate);
                        $vdate = &ctime($visit_times{$url}) if $visit_times{$url};
                        $vdate = 'Never visited ' unless $visit_times{$url};

                        chop $mdate;
                        chop $vdate;
                        print STDERR "\tvisited $vdate ($visit_times{$url}) modified $mdate\n";
                    }
                    next;
                }
                $mod = $cached_mod;
            } elsif ($cached_mod < 0) {
                $cached_cs = -$cached_mod;
            }
        }
        if ($nobots eq ':nobots') {
            push (@nobots,$url);
            print STDERR "$url known not to allow robots: skipping\n" if $options{'verbose'};
            next;
        }
        print STDERR "URL $url visited $visit_times{$url} mod $mod\n" if $options{'debug'};
        # store the max for later - used for knowing when to add emphasis
        $urls_checked{$url} = $last_checked;
        if ($last_checked < $t || time) {
            printf(STDERR "Using cached %.1f days old threshold %d [$stamp{$url}]\n",
                   (time - $last_checked) / 86400, $t) if $options{'verbose'};
            if ($mod > $visit_times{$url}) {
                push(@new_urls, $url);
```

```
                } else {
                    push(@skipped_urls, $url);
                }
                $urls_time{$url} = $mod if $mod > 0;
                next;
            }
            print STDERR "Checking $url\n" if ($options{'verbose'});
            next if ($options{'nop'});
            # need to change this to handle missing files, down hosts, etc  -- be
            # smarter about return codes.
            local($sendcmd) = sprintf("%s %d %d\n", $url, $last_checked,
                                      ($cached_mod > 0) ? $cached_mod : 0);
            print SENDURLS $sendcmd;
            printf STDERR "--> $sendcmd" if $options{'debug'} > 0;
        } else {                    # file
            local($file) = $url;
            $file =~ s"file:"";
            $file =~ s"file:~/"$ENV{'HOME'}/";
            $file =~ s"/localhost??;
            if (-e $file) {
                local(@ary) = stat(_);
                $timet = $ary[$ST_MTIME];
                $urls_time{$url} = $timet;
                $stamp{$url} = "local filesystem";
                print(STDERR "file $file exists => $timet\n") if ($options{'debug'});
                if ($timet > $visit_times{$url}) {
                    push(@new_urls, $url);
                } else {
                    push(@old_urls, $url);
                }
            } else {
                push(@inaccessible_urls, $url);
                print(STDERR "File $file does not exist\n") if ($options{'deb,'});
                next;
            }
        }
    }
;
close(SENDURLS);

while (<GETRESULTS>) {
    print(STDERR "<---$_") if $options{'debug'} > 0;
    chop;
    ($url, $mod, $status, $resp, $rest) = split(/\s+/);
    die "Bad input from back end" if $rest ne "" || $status eq "";
    &nosuch$url($url), next if !defined $urls{$url};
    if ($status eq 'ROBOTS') {
        push (@robots,$url);
        $cached_modtimes{$url} = join(:, $urls_time{$url}, $visit_times{$url}, "robots");
        next;
    }
    if ($status eq 'NOK') {
        $status{$url} = $resp if $resp ne "";
        push(@inaccessible_urls, $url);
        # don't check again until timeout
        # $urls_checked{$url} = time;
        $cached_modtimes{$url} = join(:, $urls_time{$url}, time);
        next;
    }
    if ($status ne "OK") {
        print(STDERR "Warning, bad status $status from back end, url $url\n");
    }
    if ($mod > 0) {
        local ($changed);
        if ($mod <= $visit_times{$url}) {
            push(@old_urls, $url);
            $changed = 0;
        } else {
            push(@new_urls, $url);
            $changed = 1;
        }
        $urls_time{$url} = $mod;
        $cached_modtimes{$url} = join(:, $mod, time);
        print(STDERR "OK: $url ($mod) $changed\n", $changed ? "":"un")
            if ($options{'verbose'});
        $stamp{$url} = "polled";
    } elsif ($mod < 0) {
        print(STDERR "Checksum($url):$cached_cs => %d\n", -$mod)
                if ($options{'debug'});
        ($cached_modtimes{$url} = join(:, -$new_cs, time);
        if ($cached_cs != $new_cs) {
            push(@new_chksums, $url);
        else {
```

```
            push(@old_checksums, $url);
        }
    } else {
        print STDERR "NOK, $url resp $response\n" if ($options{'verbose'});
        push(@inaccessible_urls, $url);
    }


exit if ($options{'nop'});

print OUT "<html>"."\n";
if ($options{'display_header'}) {
    chop($date = &ctime(time));
    print OUT <<EOF;
<title>Status of URLs</title>
<h1>This page generated by <a href="$w3newer_url">w3newer v$w3newer_ver</a> on $date</h1>
EOF
        }
print OUT &display_url_list;
if ($options{'display_footer'}) {
    print OUT <<EOF;
<hr>
This page generated by <a href="$w3newer_url">w3newer v$w3newer_ver</a>
a program written by $w3newer_author and modified by $w3newer_authors.
EOF
        }
print OUT "</html>"."\n";
close(OUT);

#save_checksum if ($options{'checksum'});

if ($options{'cache_modtimes'}) {
    &write_cached_modtimes($options{'cm_file'}, %cached_modtimes);
}

exit;

sub url_desc {
    local ($url) = @_;
    if ($urls{$url}) {
        return $urls{$url};
    } else {
        return $url;
    }


sub bydesc {
    &url_desc($a)<=>&url_desc($b);


sub display_url_by_month {
    local (@urllist) = @_;
    local($url,$old_month,$month,$month_no,$list,$_,$day,$wday,$wdaynum,$bold);
    local($vdate, $source);
    $list = "";
    $month = "";
    for $url (@urllist) {
        $old_month = $month;
        ($day,$month_no,$year,$wdaynum) = (gmtime($urls_time{$url}))[3,4,5,6];
        $month = $months[$month_no];
        $wday = $days[$wdaynum];
        if ($month ne $old_month) {
            $list .= "</ul>\n" if ($old_month);
            $list  .= "<h1>$month $year</h1>\n";
            $list  .= "<ul>\n";
        }
        $list  .= "<li> <nobr> <a href=\"$url\">";
        $list .= &url_desc($url);
        $list  .= "</a> ";
        $bold = !$options{'embolden'} &&
                ($urls_time{$url} > $visit_times{$url})  &&
                ($visit_times{$url} > 0);
        if (!$options{'last_visited'} && $visit_times{$url} > 0) {
            chop($vdate = &ctime($visit_times{$url}));
            $vdate = ", visited $vdate";
        } else {
            $vdate = "";
        }

        if (!$options{'timestamp_source'} && $stamp{$url} ne "") {
            $source = ", based on $stamp{$url}"
        else {
```

```
            $source = ''
        }

        $list  .= '<STRONG>' if $bold;
        $list  .= "$wday, $month $day";
        $list  .= '</STRONG>' if $bold;
        $list  .= $vdate;
        $list  .= $source;
        $list  .= "\n";

        $list .= &snapshot($url) if $options{'snapshot'};
        $list .= &snapshot($url);
        $list  .= "</nobr>\n";
    }
    $list .= "</ul>\n</ul>\n";


sub snapshot {
    local ($url) = @_;
    return '' if $url !~ 'http://';
    if ($options{'forms'}) {
        return qq'
<font size=-1>
<form method=POST action="$options{'snapshot_url'}">
<input name="url" type="hidden" value="$url">
<input name="email" type="hidden" value="$options{'from_address'}">
<select name="type">
    $option_string
</select>
<input type="submit" value="invoke">
</form>
</font>';
    } else {
        local ($a) = qq!<A HREF="$options{'snapshot_url'}?email=$options{'from_address'}&url=$url&type=!; @{[
        return qq' | $[a]Remember">Remember</A> | $[a]Diff">Diff</A> | $[a]History">History</A> ' : '';
    }
}

sub display_url_list {
    local($list) = "<!-- This page generated by w3newer $w3newer_ver by "
        ."Brooks Cutter ($w3newer_author_email) -->\n";
    if (@new_urls) {
        $list  .= "<h2>The following URLs have changed:</h2>\n";
        $list  .= &display_url_by_month(reverse sort { $urls_time{$a}<=>$urls_time{$b} } @new_urls);
    }
    if ($options{'checksum'}) {
        if (@new_chksums) {
            $list .= "<h2>The following URLs may have changed (no modification time available):</h2>\n<ul>\n";
            foreach $url (@new_chksums) {
                $desc = &url_desc($url);
                $list .= qq?<li><nobr> <a href="$url">$desc</a>\n?;
                $list .= &snapshot($url) if $options{'snapshot'};
                $list .= "</nobr>";
            }
            $list .= "</ul>\n";
        }
        if (@old_chksums) {     #
            $list  .= "<h2>The following URLs don't seem to have changed (no modification time available):</h2>\n<ul>\n";
            foreach $url (@old_chksums) {
                $desc = &url_desc($url);
                $list .= qq?<li><nobr> <a href="$url">$desc</a>\n?;
                $list .= &snapshot($url) if $options{'snapshot'};
                $list  .= "</nobr>";
            }
            $list  .= "</ul>\n";
        }
    }
    if ($options{'display_recent'} && @skipped_urls) {
        $list  .= "<h2>The following URLs were checked recently and not polled.</h2>\n<ul>\n";
        for $url (sort bydesc @skipped_urls) {
            $desc = &url_desc($url);
            $list .= qq?<li><nobr> <a href="$url">$desc</a>?;
            if ($options{'timestamp_source'} && $stamp{$url} ne '') {
                local ($age) = time - $urls_checked{$url};
                if ($age < 0) {
                    if ($options{'verbose'}) {
                        local($foo, $foo1, $foo2);
                        chop($foo=&ctime(time));
                        $foo1 = $foo;
                        chop($foo=&ctime($urls_checked{$url}));
```

```
                    $foo2 = $foo;
                    warn "$url apparently checked in the future";
                    printf(STDERR "time %s checked %s\n", $foo1, $foo2);
                }
                $age = 0;
            }

            local ($t) = &secs2str($age);
            $t =~ s/\d+s//;
            local($ago) = sprintf("%s %s",
                                  ($stamp{$url} eq "history") ?
                                  "visited" : "checked",
                                  $t);
            $list .= ", based on $stamp{$url}: $age ago)";

        $list .= &snapshot($url) if $options{'snapshot'};
        $list .= "</nobr>\n";

    $list .= "</ul>\n";


if ($options{'display_unchanged'} && @old_urls) {
    $list .= "<h3>The following URLs have not changed</h3>\n";
    $list .= &display_url_by_age(reverse sort { $urls_time{$a}<=>$urls_time{$b} } @old_urls);
}


if ($options{'display_unknown'} && @inaccessible_urls) {
    $list .= "<h3>Unable to access the following URL's:</h3>\n<ul>\n";
    for $url (sort {$urls{$a}<=>$urls{$b}} @inaccessible_urls) {
        $desc = &url_desc($url);
        $list .= qq"<li><nobr> <a href=$url>$desc</a>\n";
        if ($urls_accessed{$url} > 0) {
            ($sec,$visit) = actime($urls_accessed{$url});
            $list .= "(Last visited $visit) ";
        }
        $list .= &snapshot($url) if $options{'snapshot'};
        $list .= "</nobr>";
        if (defined($status{$url})) {
            local($why) = $w3error'RespMessage($status{$url}); #";
            if (defined($why)) {
                $list .= "<UL><LI>$why</UL>\n";
            }
        }
    }
    $list .= "</ul>\n";
}
if ($nobots) {
    $list .= "<h3>The following URLs are not accessible to robots:</h3>\n<ul>\n";
    foreach $url (@nobots) {
        $desc = &url_desc($url);
        $list .= qq"<li><nobr> <a href=$url>$desc</a>\n";
        $list .= &snapshot($url) if $options{'snapshot'};
        $list .= "</nobr>";
    }
    $list .= "</ul>\n";
}

return($list);
}

# a generic routine to figure out what kind of hotlist we're looking at.
# potentially useful if the hotlist is fed in through stdin or something.
sub load_hotlist {
    local($file,*hotlist_urls, *hotlist_dates) = @_;
    local(*desc,*times,$_);

    open(IN,$file) || die "Can't open file $file for reading : $!\n";

    $_ = <IN>;
    if ($ncsa=~/^mosaic-hotlist-format-1/) {
        &load_mosaic_hotlist_or_history(IN,1, *hotlist_dates, *hotlist_urls);
    } elsif (/^<!DOCTYPE NETSCAPE-Bookmark-file-1>/) {
        &load_netscape_bookmarks(IN, *hotlist_dates, *hotlist_urls);
    } else {
        die "Don't know how to handle hotlist with format $_";
    }
    close(IN);


sub load_mosaic_hotlist_or_history {
    local($handle,$hotlist,*times, *descs) = @_;
    local($_, $url, $desc,$time);
```

```
<$handle>                    # discard 'Default/Global'
while (eof($handle)) {
    chop($_ = <$handle>);
    if (/^(\S+)\s+(\d+)/) {
        $url = $1;
        $time = $2
    } else {
        die "Can't parse input line $_";
    }
    if ($hotlist) {
        chop($desc = <$handle>);
        $descs{$url} = $desc;
    }
    $times{$url} = &localdatetomtime($time);
}


sub load_netscape_bookmarks {
    local($handle, *times, *descs) = @_;
    local($_, $url, $desc, $time);
    local(@links, $links);

    <$handle>;                    # discard 'Default'

    @content = <$handle>;         # rest of file
    &parse_html(*links, join("", @content));
    $links = -1;
    while (defined $links[++$links]) {
        $_ = $links[$links];
        if (/^<a\s+href\s*=\s*"([^"]*)"[^>]*>$/i ||
            /^<a\s.*href\s*=\s*(\S+).*>$/i) {
            $url = $1; $desc = '';
            if (/LAST_VISIT="(.*?)"/) {
                $time = $1;
            } else {
                $time = 0;
            }
            while (defined $links[++$links]) {
                last if ($links[$links] =~ m!<\s*/\s*a\s*>!i);
                $desc .= $links[$links];
            }
            print STDERR "extract_links found URL $url time $time\n" if ($options{'debug'} > 5);
            $descs{$url} = $desc;
            $times{$url} = $time;
        } else {
            print STDERR "extract_links ignoring $_\n" if ($options{'debug'} > 5);
            next;
        }
    }
}

# Ditto for history files.
sub load_history {
    local($file) = @_;
    local(%times,$_);

    open(IN,$file) || die "Can't open file $file for reading : $!\n";

    print "Loading history file....";
    $_ = <IN>;
    if (/^ncsa-mosaic-history-format-1/) {
        &load_mosaic_hotlist_or_history(IN,0,*times);
    } elsif (/^MCOM-Global-history-file-1/) {
        %times = &load_netscape_history(IN);
    } else {
        local($fn) = $options{'debug'} ? 'die' : 'warn';
        eval "$fn \"Don't know how to handle history file $file with format\n$_\"";
    }
    close(IN);
    print "done\n";
    local($key);
    for $key (keys %times) {
        $status{$key} = "history";
    }
    return %times
}


sub load_netscape_history
```

```
    local($_,$url,$time);

    while (<$handle>) {
        chop;
        if (/(\S+)\s+(\S+)/) {
            $url = $1;
            $time = $2;
        } else {
            die "Can't parse input line $_";
        }
        $times{$url} = $time;
    }
    return \%times;
}


sub extract_links_desc {
    local($in_url) = shift(@_);
    local(@links,@tlinks,%links);
    local($ret,$url,$_,$desc,%headers,$content,$response);

    $response =
        &www'request('GET', $in_url, \%headers, "", $gl_request_timeout); #'

    if ($options{'debug'}) {
        print STDERR "---------WWW content--------\n";
        print STDERR $content;
        print STDERR "\n---------end WWW content--------\n";
    }

    &parse_html(*links,$content);
    $links = -1;
    while (defined $links{--$links}) {
        $_ = $links{$links};
        if (/^<a\s.*href\s*=\s*"([\S+)".*>$/i)
            || /^<a\s.*href\s*=\s*([\S+).*>$/i)) {
            $url = $1; $desc = '';
            while (defined $links{--$links}) {
                last if ($links{$links} =~ m!<\s*/\s*a\s*>!i);
                $desc .= $links{$links};
            }
            print STDERR "extract_links found URL $url\n" if ($options{'debug'} > 5);
            push(@ret,$url,$desc);
        } elsif (!/^\s*<img\s+src\s*=\s*"([\S+)".*>$/i)
                || /^\s*<img\s+src\s*=\s*([\S+).*>$/i)) {
            print STDERR "extract_links found IMG $1\n" if ($options{'debug'} > 5);
            push(@ret,$1,$_);
        } else {
            print STDERR "extract_links ignoring $_\n" if ($options{'debug'} > 5);
            next;
        }
    }
    return(@ret);
}

sub display_html {
    local($in_url) = shift(@_);
    print STDERR "display_html($in_url)\n" if !$options{'debug'};
    local(%links,@links,$links);
    local(@ret,$url,$_,$desc,%headers,$content,$response);
    local($day,$month_no,$year,$wdaynum);

    $response = &www'request('GET', $in_url, \%headers, "", $gl_request_timeout); #

    &parse_html(*links,$content);
    $links = -1;
    while (defined $links{--$links}) {
        if ($links{$links} =~ /^<\s*w3newers+url+"([^"]+)"\s*>$/i) {
            ($time) = (&www'stat($1,$UserAgent,$ReplyTo))[1]; #'
            unless ($time) { push(@ret, "Unknown"); next; }
            ($day,$month_no,$year,$wdaynum) = (gmtime($time))[3,4,5,6];
            $month = $months[$month_no];
            $wday = $days[$wdaynum];
            push(@ret,"$wday, $month $day $year");
        } else {
            push(@ret,$links{$links});
        }
    }
    return @ret;
```

```
sub read_cached_modtimes {
    local ($file) = @_;
    local (%times, $url, $mod, $checked, $nobots);
    open(CACHED,$file) || return ();
    while (<CACHED>) {
        ($url, $checked, $mod, $nobots, $rest) = split(/\s+/);
        die "Bad input from $file" if $rest ne "";
        $times{$url} = join(:, $mod, $checked, $nobots);
        $stamp{$url} = "previous run";
    }
    close(CACHED);
    return %times;
}

sub write_cached_modtimes {
    local ($file,%cached) = @_;
    local ($mod, $checked, $key, $nobots);
    return if !keys %cached;
    rename($file, "$file~") if -e $file;
    if (!open(CACHED,">$file")) { warn "Unable to write file $file"; return; }
    foreach $key (keys %cached) {
        ($mod, $checked, $nobots) = split(':', $cached{$key});
        print(CACHED "$key\t$mod\t$checked\t$nobots\n", $key, $checked, $mod, $nobots);
    }
    close(CACHED);
}

# ------------------ ------------------------------------------------------
# localdatetomtime() is a modification of the get_mtimefrom w3dates.pl
#
#
# Description:
#       Translate a date string to machine time (seconds since Epoch)
#
# Usage:
#       $mtime = &localdatetomtime($date);
#
#    where $date is in the following format:
#
#       "Thu Feb  3 17:03:55 1994"

    sub localdatetomtime
{
    local($_) = @_;
    local($[) = 0;
    local($day, $mn, $yr, $hr, $min, $sec, $mdate, $mtime, $mon, $maxdx);
    local(%offset) = 0;

    # Split date string
    local(@w) = split;

    ($weekday, $mn, $day, $mtime, $yr) = @w;
    $yr    -= 1900;
    ($hr, $min, $sec) = split(/:/, $mtime);

    if (!$mn || ($yr < 70)) { return 0; }

    # Translate month name to number
    $maxdx = index($Mstr, substr($mn,0,3));

    if ($maxdx  < 0) { return 0; }
    else             { $mon = $maxdx / 3; }

    # Translate to seconds since Epoch
    return (timelocal($sec, $min, $hr, $day, $mon, $yr));
}

# EOF
```

```
#!/usr/local/bin/perl

# This script is a backend to the snapshot script in my CGI directory
# It is separated primarily for purposes of debugging: it doesn't rely
# on having a CGI-environment set up and instead works just with
# command-line args.

# One of the things being done for simplicity here is to just keep
# appending to the log of <url,timestamp> tuples, instead of
# overwriting in place.

# to do:
#   locking

$debug=1;
$need_locks = 1;
$maxlength = 1024 * 512;        # 512 Kbytes
$snapshot = 'http://www1126.research.att.com/cgi-bin/nc_hands';
# this should be PWD but that may not be normalized to /home.
$snapdir = '/home/douglis/tmp/cgitest/snapshot';

{
    local($oldfh) = select(STDERR);
    $| = 1;
    select(STDOUT); $| = 1;
    select($oldfh);
}


unshift(@INC, '/home/douglis/lib/perl');
# architecture-specific library (for sys/socket.ph)
require 'arch.pl';
$arch = &arch;
if (-d ($libloc = '/home/douglis/arch/$arch/lib/perl')) {
    unshift(@INC, $libloc);
}
#
($FD_CLOEXEC, $F_SETFD) = (1, 2); # from solaris sys/fcntl.h

$lib_www_ver = '0.40';
unshift(@INC,
  $ENV{'LIBWWW_PERL'} || '/home/douglis/lib/perl/libwww-perl-$lib_www_ver',
);
$root = $ENV{'PWD'};


if (defined $ENV{'PATH'}) {
#    print STDERR "Path: $ENV{'PATH'}<br>" if $debug;
} else {
    $ENV{'PATH'} = '/home/douglis/arch/$arch/bin:/home/douglis/bin:/bin/:/usr/sbin:/usr/bin:/usr/ucb';
}

unless($ENV{'http_proxy'}) {
#    $ENV{'no_proxy'}="att.com";
     # want research.att.com to go outside
    $ENV{'no_proxy'}="tbg.att.com,ncr.com,radish.research.att.com,www1126.research.att.com,lb.att.com,ho.att.com,cb.att.com,mt.att.com,divl11.att.com";
    $ENV{'http_proxy'}="http://radish.research.att.com:8000/";
#    $ENV{'http_proxy'}="http://blue1.research.att.com:8080/";
}

require "cgi-lib.pl";
$EXIT_FN = "cleanup_tmp";
require "cgi-exit.pl";
require "cgi-alarm.pl";
require "duplexpipe.pl";
require "normalize_html.pl";
require "ctime.pl";
require "stat.pl";
require "www.pl";
require "lock.pl" if $need_locks;
#require "flush.pl";
require "printid.pl";

if ($debug) {
    $argstring = join(' ', @ARGV);
    print STDERR &printid;
```

```
require 'findprog.pl';
#($rcsdiff,$htmldiff,$filter_anchors)=&findprog('rcsdiff', 'htmldiff', 'filter_anchors');
($rcsdiff,$htmldiff,$rlog2html)=&findprog('rcsdiff', 'htmldiff', 'rlog2html');
&exit_gracefully("Unable to locate rcsdiff", 0) if !defined $rcsdiff;
&exit_gracefully("Unable to locate htmldiff", 0) if !defined $htmldiff;
&exit_gracefully("Unable to locate rlog2html", 0) if !defined $rlog2html;
#&exit_gracefully("Unable to locate filter_anchors", 0)
#    if !defined $filter_anchors;
$rcsdir = $rcsdiff;
$rcsdir =~ s:/rcsdiff$::;

$ci="$rcsdir/ci";
$co="$rcsdir/co";
$rcs="$rcsdir/rcs";

$no_hands_url =
  'http://www1126.research.att.com/~douglis/no_hands/';

$no_hands_author = <<EOF;
<A HREF="http://www1126.research.att.com/~douglis/">Fred Douglis</A>
EOF
chop($no_hands_author);
$htmldiff_author = <<EOF;
<A HREF="http://www-spr.ih.att.com/~tball/">Tom Ball</A>
EOF
chop($htmldiff_author);

# &exit_gracefully("Wrong number of arguments", 1) if $#ARGV != 2;

# need full path here because of possible BASE directive
$logo = qq!<a href="http://www1126.research.att.com/~douglis/no_hands/">
<img align="left" alt="[NO HANDS logo]"
src="http://www1126.research.att.com/~douglis/no_hands/logo.gif"></A>\n!;
$trailer = <<EOF;
</BODY>
<hr>
<FONT SIZE=3>
This page generated by <a href="$no_hands_url">NO HANDS</a>,
an <strong>AT&amp;T proprietary</strong> program written by $no_hands_author.
<br>
Any <a href="mailto:nohands\@toucan.research.att.com">comments</a> are appreciated.
$logo
</HTML>
EOF
$htmldiff_trailer = qq!, using <i><a href="http://www-spr.ih.att.com/~tball/htmldiff.html">htmldiff</a></i> (also proprietary;, wri
tten by $htmldiff_author!;


($operation, $user, $url, $userversion) = @ARGV;

@tmpfiles = ();

# assoc. array of operations and required fields.  u->URL, e->email(user),
#       w->write access, p->people
%allowable_operations = ('Remember', 'uew',
                         'Diff', 'uew',
                         'Status', 'ue',
                         'List_All', 'e',
                         'History', 'ue',
                         'Users', 'p',
                         'Remember_if_new', 'uew',
                         'All_URLs', 'p',
                         'URL_users', 'up',
                         'View_Current', 'ue'
);

&exit_gracefully("Illegal operation $operation specified", 1)
    unless defined $allowable_operations{$operation};

# print STDERR "\$allowable_operations{$operation}=\"$allowable_operations{$operation}\"\n" if $debug;

if ($allowable_operations{$operation} =~ /e/) {
    &exit_gracefully("Illegal user name", 1) if $user =~  /^\w@.]+$/;
    $lines = `people/$user`;
```

```
if (!allowable_operations{$operation} =~ /u/) {
    &exit_gracefully(sprintf("Illegal URL: '%s'", $url), 1);
    if $url !~ m!^http://[\w0.-,/\~:]+$!;
# temp until add logging
    if ($operation eq 'View_Current') {
        print "Location: $url\n\n";
        exit;
    }
    $need_rcs = 1;
} else {
    $need_rcs = 0;
}

if ($allowable_operations{$operation} =~ /p/) {
    opendir(PEOPLE, 'people') || &exit_gracefully("Unable to open directory 'people'", 0);
    local(@people) = readdir(PEOPLE);
    @times = grep(/^[a-z]/ && s!^!people/!, @people);
    print( STDERR "Checking users %s\n", join(' ', @times) if $debug > 1;
    close(PEOPLE);
}

if ($need_rcs) {
    $fn_url = &url2fn($url);
    $rcsfile = &rcsfile($fn_url);

    if (-e $rcsfile) {
        print( STDERR "%s previously checked in\n", $url if $debug > 2;
        #    &exit_gracefully("File $rcsfile not writable", 0) unless -w $rcsfile;
        $existed = 1;
    } else {
        $existed = 0;
    }
}
@users = () if $operation eq "URL_users";

if (!$operation eq "Diff" && $userversion !~ /\d+/) ||
    $operation eq "List_All" ||
    $operation eq "History" || $operation =~ /.*users/i ||
    $operation eq "Remember_if_new" || $operation eq "All_URLs") {
    foreach $times (@times) {
        if ($operation ne "URL_users") {
            undef %lastseen;
            undef %lastseen;
        }
        if (-r $times) {
            &exit_gracefully("file $root/$times not writable", 0) if ! -w "$times";
            open(TIMES, "<$times") || &exit_gracefully("Unable to open $root/$times: $!", 0);
            unless (!$need_locks || &lock(TIMES, 1)) {
                &exit_gracefully("Unable to acquire lock on $times",
                                 "A file was temporarily unavailable.  Please try again later. (You can reload this page )");
            }
            while (<TIMES>) {
                next if /^$/;
                local($turl, @when) = split(' ', $_);
                if ($operation eq "Diff" || $operation eq "History" ||
                    $operation eq "Remember_if_new" || $operation eq "URL_users") {
                    if ($turl eq $url) {
                        if ($operation eq "URL_users") {
                            local($user) = $times;
                            $user =~ s!^people/!!;
                            $lastseen{$user} = "@when";
                        } else {
                            $prevseen = $lastseen if $userversion eq "penultimate";
                            $lastseen = "@when";
                            $lastseen = join " ", @when);
                        }
                    }
                } elsif ($operation eq "List_All" || $operation eq "Users"
                         || $operation eq "All_URLs") {
                    $lastseen{$turl} = "@when";
                    $lastseen{$turl} = join(' ', @when);
                    if ($operation eq "All_URLs") {
                        # set here: use if set just 1x
                        $lastuser{$turl} = $times;
                        $lastuser{$turl} =~ s!^people/!!;
                    }
```

```
            $lastseen = $prevseen if defined $prevseen && $userversion eq 'penultimate';
            close (TIMES);
            &unlock(TIMES) if $need_locks;
            if ($operation eq 'Users') {
                local (@keys) = keys %lastseen;
                local($user) = $times;
                $user =~ s!^people/!!;
                $count{$user} = $#keys + 1;
            } elsif ($operation eq 'All_URLs') {
                for (keys %lastseen) {
                    $count{$_}++;
                }
            }
        } elsif ($operation eq 'Users') {
            print "Nothing has ever been checked in.";
            print $trailer;
            exit 0;
        }
    }
    if ($operation eq 'List_All') {
        print &Title("Pages recorded for $user");
        print "<h4>Sorted by Domain</h4><p>\n";
        print $logo;
        print "<TABLE> <TR> <TH>URL</TH><TH>Modification timestamp</TH></TR>\n"
        for (sort bydomain keys %lastseen) {
            local($url, @rest) = split(' ', $_);
            printf qq!<TR><TD><A HREF="%s?url=%s&email=%s&type=History">%s</A></TD><TD>%s</TD></TR>\n!, $snapshot, $url, $user, $_
1, $lastseen{$_});
        }
        print "</TABLE>", $trailer;
        exit 0;
    } elsif ($operation eq 'Users') {
        print &Title("Users of the NO HANDS service");
        print "<TABLE> <TR> <TH>User</TH><TH>Pages recorded</TH></TR>\n";
        for (reverse sort byusercount keys %count) {
            printf qq!<TR><TD><A HREF="%s?email=%s&type=List%20All">$_</A> </TD><TD>%d</TD></TR>\n"  $snapshot, $_, $count{$_
        }
        print "</TABLE>", $trailer;
        exit 0;
    } elsif ($operation eq 'URL_users') {
        print &Title("Users who have recorded $url");
        print $logo;
        for (sort keys %lastseen) {
            printf qq!<A HREF="%s?email=%s&type=List%20All">$_</A> (%s)<BR>\n!, $snapshot, $_, $lastseen{$_};
        }
        print $trailer;
        exit 0;
    } elsif ($operation eq 'All_URLs') {
        print &Title("Pages recorded by the NO HANDS service");
        print $logo;
        print "<TABLE> <TR> <TH>URL</TH><TH>User(s)</TH><TH>Last version</TH></TR>\n";
        for (sort byusercountURL keys %count) {
            printf qq!<TR><TD><A HREF="%s?url=%s&email=%s&type=View%20Current">%s</A></TD><TD>",
                $snapshot, $_, $user, $_;
            if ($count{$_} > 1) {
                printf qq!%d <A HREF="%s?url=%s&type=URL%20users&email=%s">users</A>!,
                $count{$_}, $snapshot, $_, $user;
            } else {
                printf qq!<A HREF="%s?type=List%20All&email=%s">%s</A>!,
                $snapshot, $lastuser{$_}, $lastuser{$_};
            }
            print "</TD>";
            local($fn) = &rcsfile(&url2fn($_));
            getversion: if (-e $fn) {
                open(RCSFILE, $fn) || warn "Can't open $fn", last getversion;
                # do we need to lock just to read 1st line?
                local($line) = <RCSFILE>;
                if ($line =~ /^head\s+(\d+\.\d+)/) {
                    printf qq!<TD><A HREF="%s?url=%s&email=%s&type=History">%s</A></TD>\n!,
                    $snapshot, $_, $user, $1;
                }
                close(RCSFILE);
            }
            print "</TR>\n";
        }
        print "</TABLE>\n$trailer";
        exit 0;
    } elsif ($operation eq 'Remember_if_new') {
```

```
        .!  defined($lastseen)) {
            $msg = <<EOF;
This <A HREF="$url">page</A> was previously saved.
<A HREF="$snapshot?url=$url&email=$user&type=History">View the history</A>
EOF
    print $msg.$trailer;
            exit 0;

    } elsif (.!defined($lastseen) && $operation eq "Diff") {
        $msg = <<EOF;
This URL has not previously been saved by you.
<UL>
<LI> <A HREF="$snapshot?url=$url&email=$user&type=Remember">Take a snapshot</A>
<LI> <A HREF="$url">View current version</A>
<LI> <A HREF="$snapshot?url=$url&email=$user&type=History">View the history</A>
</UL>
EOF
    &exit_gracefully("User tried to access a URL not previously seen",
                     $msg);
    } elsif    -e $rcsfile) {
        &exit_gracefully("The RCS history associated with the url <a href=\"$url\">$url</a> is unavailable", 1);


*   local(%headers,$content,$response);

if ($operation eq "Diff" || $operation =~ /Remember.*/ || $operation eq "History") {
  $current = "pages/$fn_url";
#  if (-e $current) {
#    unlink ($current) || &exit_gracefully("Unable to unlink $current", 0);
#  }
  if ($need_locks) {
      $lockfile = "locks/$fn_url";
      open(LOCK, ">$lockfile") || &exit_gracefully("Can't lock $lockfile", 0);
      push(@tmpfiles, $lockfile);
      unless (&slock(LOCK, 0)) {
          &exit_gracefully("Unable to acquire lock on $lockfile",
                           "A file was temporarily unavailable.  Please try again later. (You can reload this page.)");
      }
      print STDERR &printid, "Acquired lock on $lockfile\n" if $debug;
  }

  if ($operation eq "Diff" || $operation =~ /Remember.*/) {
      $alivepid = &cgi_keepalive;
      print STDERR &printid, "Keepalive pid $alivepid\n" if $debug;
      $newurl = $url;
      $response = &www'request('GET', *newurl, *headers, *content); #'

          kill 2, $alivepid;    # SIGINT
      if (!waitpid($alivepid, 0)) {
          printf STDERR "%s warning: unable to wait for child keepalive process $alivepid\n", &printid;
      }

      if (!$response =~ /^2/) {
          local($rest) = $wwwerror'RespMessage($response); #';
          if (defined($rest)) {
              $rest = ": $rest";
          }
          &exit_gracefully("Response $response during GET request$rest.", 1);
      }
      local ($len) = length($content);
      if ($len > $maxlength) {
          &exit_gracefully("The page is too large to cache. Content length is $len; maximum is $maxlength.", 1);
      }

      printf( STDERR "Get worked!!\n content:\n%s...\n",
            &substitute_entities (substr($content, 0, 256))) if $debug > 1;
      $content = &normalizeHtml($content, $newurl);
      .!  $content =~ /^Error filtering HTML:/) {
          &exit_gracefully($content, 0);
      }


      open(CURRENT, ">$current") || &exit_gracefully("Unable to open $current", 0);
      push(@tmpfiles, $current);
      print CURRENT $content;
      close(CURRENT);
      $m.dir(STDOUT)
```

```
    if  -z "$current") {
        local($len);
        if (!($len = length($content)) > 0) {
            &exit_gracefully("Content not correctly copied to $content  length $len", 0);
        }
        &exit_gracefully("" "Page is apparently empty");
    }

    printf STDERR "%s%s", &printid, `ls -l $current` if $debug;
}


.f !$operation eq "Diff") {
    open(CLD, ">$current.snap") ||
        &exit_gracefully("Unable to create  $current.snap\"", 0);
    $cmd="$cmdfiles  "$current.snap"";


    if  $userversion =~  \d/) {
        $lastseen = "-r$userversion";
    } e.se
        $lastseen = "-d $lastseen";
        $lastseen = "-d"  $lastseen;
    }
    $cmd = q{$pid = &duplexpipe(*OUT, *IN, *ERRS, ("$rcs", "-p", $lastseen, "$rcsfile"))};
    $cmd = q{$pid = &duplexpipe(*OUT, *IN, *ERRS, ("/home/douglis/bin/testcc", "-p", "$rcsfile"))};

    print STDERR "Evaluating cc cand %s\n", $cmd if $debug > 2;
    eval $cmd;
    if ($@) {
        &exit_gracefully($@, 0);
    }
    while (<IN>) {
        print OLD;
        print STDERR "$... " if $debug > 2;
    }
    print STDERR "\n" if $debug > 2 && $. > 0;
    close IN;
    print STDERR &printid, "Error $? from close(IN)\n" if $? && $debug;
    close OLD;
    close(OUT);
    print STDERR &printid, "Error $? from close(OUT)\n" if $? && $debug;
    if ($debug) {
        local($first) = 1;
        while(<ERRS>) {
            print STDERR &printid, "Messages from cc:\n", undef $first if defined $first;
            print STDERR;
        }
    }
    close (ERRS);
    print STDERR &printid, "Error $? from close(ERRS)\n" if $? && $debug;
    if !(waitpid($pid, 0)) {
        print STDERR &printid, "warning: unable to wait for child process $pid\n";
    } elsif ($?) {
        print STDERR &printid, "$0: warning: cc returned status $?\n";
    } elsif !$debug) {
        print STDERR &printid, "$0: cc returned OK status\n";
    }

    if (-z "$current.snap") {
        local($ls  = `/bin/ls -l "$current.snap"`;
        &exit_gracefully($ls, "Snapshot of page is apparently empty");
    }

    if (-s "$current.snap" && -s "$current" &&
        "system("/bin/cmp", "-s", "$current.snap", "$current"")) {
        print "No differences encountered<br>\n";
        $diffs = 0;
    } else {
        $diffs = 1;
        $alivepid = &cgi_keepalive;
        print STDERR &printid, "Keepalive pid $alivepid\n" if $debug;

        $trailer =~ s/.written by $no_hands_author\../$1$htmldiff_trailer./;

        open DIFF, "$htmldiff $current.snap $current $current.diff|") ||
            &exit_gracefully("Unable to invoke htmldiff", 1);
```

```
        push(@tmpfiles, "$current.diff");
        if ($debug) {
            local($first) = 1;
            while(<DIFF>) {
                print STDERR &printid, "Messages from htmldiff:\n", undef $first if defined $first;
                print STDERR;
            }
        }
        if ($debug> 2) {
            print STDERR "Calling close(DIFF)\n";
        }
        close(DIFF);
        if ($debug) {
            print STDERR &printid, "Error $? from close(DIFF)\n" if $?;
          elsif ($debug > 2) {
            print STDERR &printid, "close(DIFF) OK\n";
        }
        kill 2, $alivepid;      # SIGINT
        if (!waitpid($alivepid, 0)) {
            print STDERR &printid, "warning: unable to wait for child keepalive process $alivepid\n";
        }
        if (-e "$current.diff") {
            if (-s _) {
                $content = `cat $current.diff`;
                $content =~ s!</html>!!i;
                print $content;
            } else {
                print "No differences encountered<br>\n";
                $diffs = 0;
            }
        } else {
            &exit_gracefully("No diff file $current.diff created", 0);
        }
    }
    if ($diffs) {
        $trailer =~ s/page/modified page/;
    }
    $moved = qq!(Note that the URL previously provided, <A HREF="$url">$url</A>, moved here.)!; if $newurl ne $url;
$maybe = qq! You should view
the current version <strong>directly</strong> if you want <a
href="http://www1126.research.att.com/~douglis/track_urls/">volunteer</a>
to know the page was seen.
<li> <A HREF="$snapshot?email=$user&url=$url&type=Remember">Remember current version</A>.<br>! unless $user eq 'trustchanges';
    $trailer .= <<EOF;
<ul>
<li> <A HREF="$newurl">view current version</A>. $moved
$maybe
<li> <A HREF="$snapshot?email=$user&url=$url&type=History">See the version history</A>.<br>
</ul>
</HTML>
EOF
    print $trailer;
} elsif ($operation =~ /Remember.*/) {
    if (!$existed) {
        if ($debug) {
            open (SAVEERR, ">&STDERR");
            open (SAVEOUT, ">&STDOUT");
            pipe (RCSERR, RCSOUT) || &exit_gracefully("Can't create pipe", 0);
            close(STDERR);
            open(STDERR, ">&RCSOUT");
            open(STDOUT, ">&RCSOUT");
        } else {
            fcntl(STDOUT, $F_SETFD, $FD_CLOEXEC) ||
                &exit_gracefully("Unable to set close-on-exec flag for stdout",
                                 0);
            fcntl(STDERR, $F_SETFD, $FD_CLOEXEC) ||
                &exit_gracefully("Unable to set close-on-exec flag for stderr",
                                 0);
        }
        system("$rcs -l $rcsfile") && &exit_gracefully("Unable to lock $rcsfile", 0);
        if ($debug) {
            close(RCSOUT);
            close(STDERR);
            close(STDOUT);
            open(STDERR, ">&SAVEERR");
            open(STDOUT, ">&SAVEOUT");
            close(SAVEERR);
```

```
            local ($warn) = 0;
            while (<RCSERRS>) {
                if (!$warn) {
                    print STDERR &printid, "Message from rcs:\n";
                    $warn = 1;
                }
                print STDERR;
            }
        } else {
            fcntl(STDOUT, $F_SETFD, 0) ||
                &exit_gracefully("Unable to clear close-on-exec flag for stdout",
                                 0);
            fcntl(STDERR, $F_SETFD, 0) ||
                &exit_gracefully("Unable to clear close-on-exec flag for stderr",
                                 0);
        }
    }
    @tmp = ("$co", "$current");
    if (defined $headers{'last-modified'}) {
        $date = "$headers{'last-modified'}";
        # check for a certain style RCS doesn't like, as in:
        # Last-modified: Friday, 16-May-95 13:19:21 GMT
        if ($date =~ /^(\w\w\w)(\w+), (\d\d)-(\w+)-(\d\d)/) {
            $date = "$1, $2 $3 19$4 $5";
        }
        splice(@tmp, 1, 1, "-d$date");
    } else {
        $date = &time2str(time);
        chop($date);
    }

    # not clear if we really need to send data to rcs -- leave as-is for now.
    #   $current2 =~ s/([^\@%\$])/\\$1/g;
    #   local($cmd) = qq!\$pid = &duplexpipe("OUT", "IN", "ERRS", \"$current2\");!;
    local($cmd) = qq!\$pid = &duplexpipe("OUT", "IN", "ERRS", \@tmp);!;
    printf STDERR "Evaluating command %s\n", $cmd if $debug > 2;
    eval $cmd;
    if ($@) {
        &exit_gracefully($@, 0);
    }
    if (!$existed) {
        local($title);
        printf OUT "This is a snapshot of a page, URL %s\n", $url;
        # would be simpler in perl5...
        if ($title = &html_title($content)) {
            printf OUT "<br><i>Title: %s</i>\n", $title;
        }
    } else {
        printf OUT qq!A snapshot made by <a HREF="$snapshot?email=%s&type=List_All>%s</a>\n!, $user, $user;
    }
    close(OUT);
    local ($warn) = 0;
$unchanged = 0;
    while (<IN>) {
        if ($debug) {
            if (!$warn) {
                print STDERR &printid, "Message from ci stdout:\n";
                $warn = 1;
            }
            print STDERR;
        }
        if (/^file is unchanged; reverting/) {
            $unchanged = 1
        }
    }
    $warn = 0;
    while (<ERRS>) {
        if ($debug) {
            if (!$warn) {
                print STDERR &printid, "Message from ci stderr:\n";
                $warn = 1;
            }
            print STDERR;
        }
    }
    close(IN);
    close(ERRS);
```

```
    if (!waitpid($pid, 0)) {
        print STDERR &printid, "warning: unable to wait for child process $pid\n";
    } else {
      STATUS: {
          local($_) = $?;
          /^[12]0[15]/ && do {    # an ok status?
              print STDERR &printid, "warning: exited with a status of $?\n"
                  if $debug;
              last STATUS;
          };
          /^05/ && last STATUS;
          &exit_gracefully("ci exited with a status of $?", 1);
      }
    }

    &exit_gracefully("Problem with 'times' file", 0) if $times != 0;
    $times = $times[0];
    &exit_gracefully("file $root/$times not writable", 0) if ! -w "$times";

    if ($unchanged) {
        $status = "Note, file was unchanged from previous check-in.";
    } else {
        $status = "Check in was successful."
    }
    print qq!$status  You can view the <A HREF="$url">current version</A> or see the <A HREF="$snapshot?email=$user&...&type=...">
<history/">version history</A>.\n!;

    print $trailer, "<p>";
    0; ?(TIMES, ">>$times") || &exit_gracefully("Unable to open $root/$times: $!", 0);
    unless (!$need_locks || &lock(TIMES, 0)) {
        print STDERR &printid, "Unable to acquire lock on $times... writing anyway.\n";
    }
    printf TIMES "%s %s\n", $url, $date;
    close (TIMES);

    &cleanup_tmp;
    exit 0;
} elsif ($operation eq "List_All") {

} elsif ($operation eq "History") {
    local($cmd) = "/usr/sbin/perl $rlog2html $snapdir/$current $url $user";
    $cmd =~ s/([\@%\$])/\\$1/g;
    printf STDERR "%sInvoking %s\n", &printid, $cmd if $debug;
    open(RLOG, "$cmd|") ||
        &exit_gracefully("can't invoke $rlog2html", 0);
    while (<RLOG>) {
        print $_;
    }
    close(RLOG);
    if ($?) {
        if ($debug) {
            print STDERR &printid, "Error $? from close(RLOG)\n" ;
        }
        print "There was an error invoking rlog.";
    } else {
        printf STDERR "close(RLOG) OK\n" if $debug > 1;
        printf qq!You can see a
<A HREF="%s?url=%s&type=URL%20user&email=%s">list of users</A> who have recorded this page.<br>\n!, $snapshot, $url, $user
        printf qq!Take a <A HREF="%s?url=%s&type=Remember&email=%s">snapshot</A> of the current page.  (You will be added to
<the list of users if you are not already there.)<br>\n!, $snapshot, $url, $user unless $user eq "truechanges";
    }
    print $trailer;
}

&cleanup_tmp;

# taken from html-parse (and reversed)
sub substitute_entities {
    local ($string) = @_;
    $string =~ s/&/&amp;/g;
    $string =~ s/</&lt;/g;
    $string =~ s/>/&gt;/g;
    $string =~ s/\"/&quot;/g;
    $string;
}
```

```
sub htmldiff_waiting {
#    print STDOUT " ";
#    print STDERR "Alarm...\n";
#    $SIG{'ALRM'} = 'htmldiff_waiting';
#    alarm($cgi_keepalive);
#}

sub byusercount {
    $count{$a} <=> $count{$b};
}

sub Title {
    local ($t) = @_;
    return "<HTML><TITLE>$t</TITLE>\n<BODY><H1>$t</H1>\n";
}

sub cleanup_tmp {
    if ($#tmpfiles >= -1 && $debug) {
        print STDERR &printid, "No temporary files to clean up.\n";
    }
    for (@tmpfiles) {
        if (-e $_) {
            print STDERR &printid, "Unlinking $_ ", `ls -l $_` if $debug;
            unlink($_) || warn "Can't unlink $_: $!";
        } else {
            print STDERR &printid, "$_ doesn't exist, ...\n" if $debug;
        }
    }
}


sub bydomain {
    $domainkey{$a} = &domainkey($a) unless defined ($domainkey{$a});
    $domainkey{$b} = &domainkey($b) unless defined ($domainkey{$b});
    return $domainkey{$a} cmp $domainkey{$b};
}
sub domainkey {
    local($url) = @_;
    local ($scheme,$address,$port,$path,$query,$frag) = &wwwurl'parse($_url); #'
    unless($port) {
        $port = 80;
    }
    $address =~ s/([.])/\l$1/g;   # lower case - better way?  where's my perl book?
    local (@addr) = split(/\./, $address);
    return sprintf("%s://%s:%d%s", $scheme, join('.', reverse(@addr)),
                                            $port, $path);
}
sub byusercountURL {
    # reverse order of count, then normal order of URL within count.
    printf STDERR "%s(%d) v %s(%d)\n", $a, $count{$a}, $b, $count{$b}
       if $debug > 3;
    local($res);
    if ($count{$a} != $count{$b}) {
        $res = $count{$b} <=> $count{$a};
        printf STDERR "\tCount: $res\n" if $debug > 3 ;
    } else {
        $res =  &bydomain;
        printf STDERR "\tURL: $res\n" if $debug > 3;
    }
    return $res;
#    local($a,$b) = ($count{$a}, $count{$b});
}
sub urlfn {
    local($url) = @_;
    local($fn_url) = $url;
    $fn_url =~ s'^http://';     # strip
    $fn_url =~ s'[^-.]+'%'g;    # flatten
    printf STDERR "Using file name \"%s\" for %s\n", $fn_url, $url if $debug > 2;
    return $fn_url;
}

sub rcsfile {
    local($fn) = @_;
    return "pages/RCS/$fn,v";
```

```
* Local Variables: *
* mode: perl *
* End: *
```

Oct 23, 1995 17:19:59 — /home/tball/douglis-tmp/cgi-bin/no_hands.cgi — king — 16:17 Oct 23 — 1 / 3

```
# -bin sh -- # a comment to keep perl from being confused

eval 'exec perl -S $0 ${1+"$@"}'
    if 0;

push (@INC, "/home/douglis/lib/perl");

$debug = 0;

require "cgi-lib.pl";
require "cgi-exit.pl";

$root = "/home/douglis/tmp/ceitest.snapshot";
$setuid = "/home/douglis/bin/nohandsEE";


    local($oldfh) = select(STDERR);
    $| = 1;
    select(STDOUT); $| = 1;
    select($oldfh);


chdir $root     &exit_gracefully("Error in chdir($root): $!" 0);

umask 0;                        # set all access bits on

@allowable_operations = ("Remember", "Diff", "Status", "List All", "History", "Users", "Remember if new", "All URLs", "URL users", "View Curre►
◄nt");


if (&MethGet) {
    &ReadParse(*input);
    if (!defined($input{'type'})) {
        if (defined($input{'email'})) {
            $emailmsg = "";
        } else {
            $input{'email'} = "$ENV{'REMOTE_USER'}\@$ENV{'REMOTE_HOST'}";
            $emailmsg = "(<i>insert your username before your hostname</i>)";
        }

        $option_string = "";
        for (@allowable_operations) {
            $option_string .= "<option> $_ ";
        }

        print &PrintHeader;
        print qq'
<html>
<title>NO HANDS</title>
<h2><IMG ALIGN=middle SRC="/logos/attbllogo.gif" >
Software and Systems Research</h2> <hr>
<body>
<H2>NO HANDS</H2>
<img align="right" alt="[NO HANDS logo]" src="/~douglis/no_hands/logo.gif">
<h2>Using This Form</h2>
This form is used to interact with the <a
href="/~douglis/no_hands/">NO HANDS</a> facility.  You can remember
what a page pointed to by a URL looks like, so you can return to it
later and see how it has changed.  (In Netscape, you can enter the URL
easily by holding down the right button over a link and selecting
<em>Copy this link location to clipboard</em>.  Note that in Netscape
1.1N, if you double-click on the URL in the <em>Location</em> line of
the page you want to track, when you come back to this page and try to
paste the URL nothing will happen.)  You may view the differences
between the most recent version you saw and the current version, or
see the history of past versions of the page (not only the ones you
saved away.  You may also see all URLs you have saved away, or see information about other users of the facility.
<P>
Note that currently there is no protection: anyone can use any
    "email address" and can view each other's information.
<p>
Some documentation about the <a href="/~douglis/no_hands/">NO HANDS</a>
facility and this <a href="/~douglis/no_hands/help.html">form</a> in
particular are available.  NO HANDS works best in conjunction with an
automatic <A HREF="/~douglis/track_urls/">notification system</A> that tells you that a page has changed and
provides the URL to access NO HANDS for that page.
<p>
<FORM METHOD=POST>
<pre>
```

```
<p> URL            <input name="url" type="text" size=40>
<p> Email address  <input name="email" type="text" value = $input{'email'} size=40> $emailmsg
<p> Operation      <select name="type">
    $option_string
        </select> .
</pre>
<p>
<input type="submit">
<input type="reset">
</form>
<P><i>Under construction</i><br>

        print &PrintTrailer;
        exit;

  else
    &ReadParse *input;

$extra = "";
#    print "Not yet implemented  <p>", &PrintVariables(%input);
unless (-w 'people' && -w 'pages' && -d 'people' && -d 'pages') {
#       if (open(TEST, ">testfile")) {
#           printf TEST `date`;
#           printf TEST "This test succeeded.\n";
#           local ($foo);
#            ($foo = $foo) =~ s/home/~/home/;
#           printf "<p> Successfully wrote <A HREF="file %s/testfile">testfile</A>.\n", $foo;
#       else
#           printf "Unable to open testfile\n";
#
    &exit_gracefully("cgitest/people or cgi/pa is not writable directory", 0);

if (!defined $input{'email'} ||
    !defined $input{'type'} || $input{'email'} eq "") {
    &exit_gracefully("CGI script invoked without sufficient POST parameters, email=$input{'email'}, type=$input{'type'}", 1);
}
undef $where;
for (0 .. $#allowable_operations) {
    $where = $_, last if $allowable_operations[$_] eq $input{'type'};

&exit_gracefully("Illegal operation $input{'type'} specified", 1)
    unless defined $where;
unless ($input{'type'} eq "View Current") {
    print &PrintHeader;

if ($input{'type'} eq "Remember") {
    printf "<Title>Remember a URL</Title>\n";
    printf "<H1>Remember URL: %s</H1>\n", $input{'url'};
}
unless ($input{'email'} =~ /^\w[\w@.]*$/ || $input{'type'} eq "Users") {
    &exit_gracefully("Script terminated due to abnormal input",
                   "email address must contain only alphanumerics, <kbd>.</kbd>, <kbd>@</kbd>, or <kbd>-</kbd> and may not start with a <kbd>
-</kbd>");
}
if ($input{'type'} eq "Remember" || $input{'type'} eq "Diff" ||
    $input{'type'} eq "Status") {
    unless ($input{'url'} =~ m?^http://[\w\.\@-/\~:]+$?) {
        &exit_gracefully("Script terminated due to abnormal input",
        sprintf("%s (offending URL: \"%s\")", "URL is restricted to <kbd>http://</kbd> followed by alpanumberics or <kbd>/ @-~: </kbd>",
                $input{'url'}));
    }

if ($input{'type'} eq "Diff" && defined($input{'version'})) {
    $extra = $input{'version'};
    if ($extra ne 'penultimate' && $extra !~ /^\d+\.\d+$/) {
        &exit_gracefully("Version for diff specified as $extra not acceptable", 1);


if ($debug) {
    open(STDERR, ">&STDOUT");
    printf "<PRE>\n";

$input{'type'} =~ s/ /_/g;
exec($backend, $input{'type'}, $input{'email'}, $input{'url'}, $extra);
print "<strong>Error processing command.</strong>\n"
print &PrintTrailer
```

```
# Local Variables:
# mode: perl
# end:
```

```
#!/bin/sh -- # a comment to keep perl from being confused

# This script invokes ctaldiff or rcsdiff on multiple versions of a
# file. It can either run on a local file (file=user:file) or a
# snapshot of a file (file=http://...). These are treated somewhat
# differently

# Actually, use diff instead of rcsdiff, and check out by hand, to
# keep the diff code itself the same regardless.

eval 'exec perl -S $0 ${1+"$@"}'
    if 0;

$debug = 1;
$need_locks = 1;
$snapshot = 'http://www1126.research.att.com/cgi-bin/nc_names'

unshift (@INC, '/home/douglis/lib/perl');
# architecture-specific library; for sys/socket.ph
require 'arch.pl';
$arch = &arch;
if (-d $libloc = "/home/douglis/arch/$arch/lib/perl")
    unshift(@INC, $libloc);
}

# What version are you using? (set this if using $olderver_dir)
$lib_www_ver = '0.40';
unshift(@INC,
  $ENV{'LIBWWW_PERL'} || "/home/douglis/lib/perl:/www-perl-$lib_www_ver",
);

if (defined $ENV{'PATH'} && $ENV{'PATH'}) {
#    print STDERR "Path: $ENV{'PATH'}<br>" if $debug;
} else {
    $ENV{'PATH'} = '/home/douglis/arch/$arch/bin:/home/douglis/bin:/bin:/usr/sbin:/usr/bin:/usr/ucb';
}

unless($ENV{'http_proxy'}) {
#    $ENV{'no_proxy'}='att.com';
     # want research.att.com to go outside
    $ENV{'no_proxy'}='bu.att.com,ncr.com,radish.research.att.com,www1126.research.att.com,sb.att.com,bo.att.com,cs.att.com,nt.att.com';
    $ENV{'http_proxy'}='http://radish.research.att.com:8000/';
}


require 'cgi-lib.pl';
$EXIT_FN = 'cleanup_tmp';
require 'cgi-exit.pl';
require 'cgi-alarm.pl';
require 'cgi-canon.pl';
require 'normalize_html.pl';
require 'hostname.pl';
require 'www.pl';
require 'lock.pl' if $need_locks;
require 'printid.pl';
require 'ctime.pl';

select(STDOUT); $| = 1;

print &PrintHeader;

# architecture-specific library (for sys/socket.ph)
require 'arch.pl';
$arch = &arch;
if (-d "$libloc = "/home/douglis/arch/$arch/lib/perl") {
    unshift(@INC, $libloc);



require 'finddiff.pl';
$rcsdiff = &finddiff('rcsdiff', 'GNUdiff', 'diff');
exit_carefully("Unable to locate rcsdiff in \$PATH=$ENV{'PATH'}", 0) if !defined $rcsdiff;
```

```
&exit_gracefully("Unable to locate htmldiff.\n\tPATH=$ENV{'PATH'}", 0) if !defined $htmldiff;
&exit_gracefully("Unable to locate diff.\n\tPATH=$ENV{'PATH'}", 0) if !defined $diff;
$rcsdir = $rcsdiff;
$rcsdir =~ s'/rcsdiff$';
$cgi="$rcsdir/cgi";

$SNAPHOME = $ENV{'SNAP'} || "/home/douglis/tmp/cgitest/snapshot";
$SNAP = "$SNAPHOME/pages";
$LOCK = "$SNAPHOME/locks";

#print STDERR "rcsdiff cgi entered\n";

$snapshot_author = <<EOF;
<A HREF="http://www.research.att.com/orgs/ssr/people/douglis/">Fred Douglis</A>
EOF
chop($snapshot_author);
$htmldiff_author = <<EOF;
<A HREF="http://www-spr.in.att.com/~tball/">Tom Ball</A>
EOF
chop($htmldiff_author);

$trailer = "<HR><FONT SIZE=-1>This page generated by a CGI script written by $snapshot_author "
$htmldiff_trailer = qq", using <i><a href="http://www-spr.in.att.com/~tball/htmldiff.html">htmldiff</a></i> written by $htmldiff_author";

$html = 0;
if (&ReadParse()) {             # this one's done with POST
    &ReadParse(*input);

    exit 1 if !$input{'diff'} || "diff";
    delete $input{'diff'};

    exit 1 if !defined($input{'file'});
    $file = $input{'file'};
    delete $input{'file'};
    if (defined $input{'realfile'}) {
        $realfile = $input{'realfile'};
        if ($realfile !~ m'^$SNAP/([\w\.\#\$\-,]+)$') {
            print "Invalid file specified: %s", $realfile;
            exit 1
        }
        $base_fn = $1;
        LOCK: {
            if ($need_locks) {
                print STDERR &printid, "trying to lock $base_fn\n" if $debug;
                open(LOCK, ">>$LOCK/$base_fn") ||
                    print STDERR &printid, "Warning: unable to open $LOCK/base_fn.\n", last LOCK;
                &slock(LOCK, 1) || print STDERR &printid, "Warning: unable to acquire lock on $LOCK/$base_fn.\n", last LOCK;
            }
        }
        delete $input{'realfile'};
        $html = 1;
        $www = 1;
        if (defined $input{'user'}) {
            $user = $input{'user'};
            delete $input{'user'};
        } else {
            $user = UNKNOWN;
        }
    } else {
        $www = 0;
        $realfile = &cgi_canon($file);
        if ($file =~ /\.html$/) {
            $html = 1;
        }
    }
    $tmp = "/tmp/rcsdiff.cgi.$$";
    $output = "/tmp/rcsdiff.cgi.diff.$$";
    @tmpfiles = ($tmp, $output);
    &cleanup_tmp;

    $versions = 0;
    $verstr = "";
    local(@versions);
    for $v (sort {$a<=>$b} keys %input) {
        push(@versions, $v);
```

```
}
if ($#versions != 1) {
    printf "Must specify exactly two versions to diff  Got %s\n"
        join(' ', @versions);
    exit;
}
@files = ();
for $v (@versions) {
    if ($html || $www) {
        # check out temporarily
        if ($v eq 'current') {
            if ($www) {
                $alivepid = &cgi_keepalive;
                print STDERR &printid, "Keepalive pid $alivepid url $file\n" if $debug;
                $origurl = $file;
                $response = &www'request('GET', $file, *headers, *content); #

                kill 2, $alivepid; # SIGINT
                if (-1==waitpid($alivepid, 0)) {
                    printf STDERR "%s warning: unable to wait for child keepalive process $alivepid\n", &printid;
                }

                if ($response !~ /^2/) {
                    local($rest) = $www'error('ResponseMessage:$response);
                    if (defined($rest)) {
                        $rest = ": $rest";
                    }
                    &exit_gracefully("Response $response during GET request$rest", 1);
                }
                print STDERR &printid, "Normalizing URL $file...\n" if $debug;
                $content = &normalizehtml($content, $file);
                if ($content =~ /^Error filtering HTML:/) {
                    &exit_gracefully($content, 0);
                }
                $tmp = pop(@tmp);
                open(TMP, ">$tmp") ||
                    &exit_gracefully("Can't open $tmp", 0);
                print TMP $content;
                close(TMP);
                push (@files, $tmp);
            } else {
                push (@files, $realfile);
            }
        } else {
            $tmp = pop(@tmp);
            system("$co -p$v $realfile > $tmp") &&
                &exit_gracefully("Error running $co -p$v $realfile > $tmp: $!", 0);
            push (@files, $tmp);
        }
    } else {
        $verstr .= "-r$v " if $v ne 'current';
    }
}

if (-s $files[0] == -s $files[1] &&
    'system("/bin/cmp", $files[0], $files[1])') {
    print "<h1>No differences encountered</h1><br>\n";
} elsif ($html || $www) {
    $filelist = join(' ', @files);
    if ($html) {
        $trailer =~ s/\.$/$htmldiff_trailer./;
        open(DIFF, "$htmldiff $filelist $output 2>&1 |") ||
                &exit_gracefully("Can't invoke $htmldiff: $!", 0);
          while(<DIFF>) {
                print STDERR $_ if $debug;
          }
          close(DIFF);
          print STDERR "Error $? from close(DIFF)\n" if $? && $debug;
          open(DIFF, "<$output") || &exit_gracefully("Can't open temporary output", 0);
    } else {
        open(DIFF, "$diff $filelist|") ||
                &exit_gracefully("Can't invoke $diff: $!", 0);

    }  if $html) {
        $content = join("\n", <DIFF>), "\n";
```

```
            close(DIFF);
            ;; ($www) {
                local($host) = (gethostbyname(`hostname`))[0];
                $url = "http://$host/$file";
                print &normalize_textblock($content, $url);
            } else {
                print $content;
            }
            print $trailer;
            if ($www) {
                local($remember) = qq(<li> <A HREF="$scripturl&type=remember">Remember current version</A> <p>);
                if ($user eq "nochanges") { # special case for me
                    $remember = "";
                }
                print <<EOF;
<ul>
<li> <A HREF="$scripturl">view current version</A>
$remember
</ul>
EOF
            }
        } else {
            print "<HTML><TITLE>rcsdiff -r $file</TITLE><BODY><PRE>";
            open(RCSDIFF, "$diff -r $version $realfile |") || exit 1;
            $lines=0;
            while (<RCSDIFF>) {
                # de-mtability !!    what else is missing?
                s/&/&amp;/g;
                s/</&lt;/g;
                s/>/&gt;/g;
                print;
                $lines++;
            }
            print "<h2>No differences encountered</h2>\n" if !$lines;
            print "</PRE></BODY>$trailer</HTML>\n";
        }
    } else {
        print "<HTML><TITLE>rcsdiff -r $version $file</TITLE>\n<BODY><PRE>\n";
        open(RCSDIFF, "$diff -r $version $realfile|") || exit 1;
        $lines=0;
        while (<RCSDIFF>) {
            # de-mtability !!    what else is missing?
            s/&/&amp;/g;
            s/</&lt;/g;
            s/>/&gt;/g;
            print;
            $lines++;
        }
        print "<h2>No differences encountered</h2>\n" if !$lines;
        print "</PRE></BODY>$trailer</HTML>\n";
    }
    &cleanup_tmp;
}

sub cleanup_tmp {
    if ($#tmpfiles >= 0 && $debug) {
        print STDERR &printid "No temporary files to clean up.\n";
    }
    for (@tmpfiles) {
        if (-e $_) {
            print STDERR &printid "Unlinking $_ ", (-l $_ ? "symlink" : "file"), "\n";
            unlink($_) || warn "Can't unlink $_: $!";
        } else {
            print STDERR &printid "$_ doesn't exist -- " if $debug;
        }
    }
}

sub myversion {
    return -1 if $a eq "current";
    return 1 if $b eq "current";
    $a <=> $b;
```

Oct 23, 1995 17:20:09 — /home/tball/douglis-tmp/cgi-bin/snapshot.cgi — king — 16:17 Oct 23 — 1 / 3

```
#  $Id: $ -- a comment to keep perl from being confused

eval "exec perl -S $0 ${1+"$@"}"
    if 0;

push @INC, "/home/douglis/lib/perl";

$debug = 1;

require "cgi-lib.pl";
require "cgi-exit.pl";

$tmp = "/home/douglis/tmp/cgitest-snapshot";
$archive = "/home/douglis/bin/nonHandsTest";  # new version


    open(SOLDIN, "<&STDIN");
    $| = 1;
    open(STDOUT, ">> $tmp");
    open(STDIN, "<&SOLDIN");


chdir $tmp  || &exit_gracefully("Error in chdir($tmp): $!");

umask 0;                        # set all access bits on

@allowable_operations = ("Remember", "Diff", "Status", "List All", "History", "Users", "Remember if new", "All URLs", "URL users", "Vie. Query");

if ($ENV{'REQUEST_METHOD'} eq "GET") {
    &ReadParse(*input);
    if (defined($input{'type'})) {
        if (!defined($input{'email'})) {
            $emailmsg = "";
        } else {
            $input{'email'} = "$ENV{'REMOTE_USER'}\@$ENV{'REMOTE_HOST'}";
            $emailmsg = "<i>Insert your username before your hostname</i>";
        }

        $option_string = "";
        for (@allowable_operations) {
            $option_string .= "<option> $_ ";
        }
        print &PrintHeader;
        print qq!
<html>
<title>NO HANDS</title>
<h2><IMG ALIGN=middle SRC="/logos/attbllogo.gif" >
Software and Systems Research</h2> <hr>
<body>
<H2>NO HANDS</H2>
<img align="right" alt="[NO HANDS logo]" src="/~douglis/no_hands/logo.gif">
<h2>Using This Form</h2>
This form is used to interact with the <a
href="~douglis/no_hands/">NO HANDS</a> facility.  You can remember
what a page pointed to by a URL looks like, so you can return to it
later and see how it has changed.  (In Netscape, you can enter the URL
easily by holding down the right button over a link and selecting
<em>Copy this link location to clipboard</em>.  Note that in Netscape
UNIX, if you double-click on the URL in the <em>Location</em> line of
the page you want to track, when you come back to this page and try to
paste the URL nothing will happen.)  You may view the differences
between the most recent version you saw and the current version, or
see the history of past versions of the page (not only the ones you
saved away).  You may also see all URLs you have saved away, or see information about other users of the facility.
<p>
Note that currently there is no protection: anyone can use any
      email address  and can view each other's information.
<p>
More documentation about the <a href="/~douglis/no_hands/">NO HANDS</a>
facility and this <a href="/~douglis/no_hands/help.html">form</a> in
particular are available.  NO HANDS works best in conjunction with an
automatic <A HREF="~douglis/track_urls/">notification system</A> that tells you that a page has changed and
provides the URL to access NO HANDS for that page.
<p>
<form method=POST>
<pre>
<b> URL            <input name="url" type="text" size=40>
<b> Url.. address  <input name="email" type="text" value = $input{'email'}, size=40> $emailmsg
```

```
<t> Operation:      <select name="type">
    $option_string
        </select>
</pre>
<p>
 <input type="submit">
<input type="reset">
</form>
<p>
Note that this page is the <b>development</b> version of the facility.  If
you have trouble with it you may find a more <a href="/cgi-bin/no_hands">
stable version</A>.
<!--<p><i>Under construction</i><br>-->
    }
        print &PrintTrailer;
        exit

  else
    &ReadParse(*input

$extra = "
#    print "Not yet implemented  <p>", &PrintVariables(%input);
#unless !-e 'people' && -w 'pages' && -d 'people' && -d 'pages') {
#      if (open(TEST  ">testfile")) {
#          print TEST `date`
#          print TEST "This test succeeded.\n"
#          local ($foo);
#          ($foo = $root) =~ s!/home/!/home1/!;
#          print "<p>Successfully wrote <A HREF=\"file:%s/testfile\">testfile</A>  \n", $foo;
#      } else {
#          print "Unable to open testfile\n";
#      }
    &exit_gracefully("cgitest/people or cgi/pages not writable directory", 0);
}
if (!defined $input{"email"} ||
    !defined $input{"type"} || $input{"email"} eq "") {
    &exit_gracefully("CGI script invoked without sufficient POST parameters. email=$input{'email'}, type=$input{'type'}", 1);
}

undef $where;
for ($i = 0; $i < $#allowable_operations; $i++) {
    $where = $i, last if $allowable_operations[$i] eq $input{"type"};
}
&exit_gracefully("Illegal operation $input{'type'} specified", 1)
    unless defined $where;
unless ($input{'type'} eq "View Current") {
    print &PrintHeader;

if ($input{"type"} eq "Remember") {
    print "<Title>Remember a URL</Title>\n";
    printf "<H1>Remember URL: %s</H1>\n", $input{"url"};

unless ($input{"email"} =~ /^\w[\w0-.]*$/ || $input{"type"} eq "Users") {
    &exit_gracefully("Script terminated due to abnormal input",
                     "email address must contain only alphanumerics, <kbd>.</kbd>, <kbd>@</kbd>, or <kbd>-</kbd> and may not start with a <kbd>
<kbd>-</kbd>");
}
if ($input{"type"} eq "Remember" || $input{"type"} eq "Diff" ||
    $input{"type"} eq "Status") {
    unless ($input{"url"} =~ m!^http://[\w\.\$!~/\-:]+$?) {
        &exit_gracefully("Script terminated due to abnormal input",
        sprintf ("%s (offending URL: \"%s\")", "URL is restricted to <kbd>http://</kbd> followed by alphanumerics or <kbd>/.$~-_:</kbd>",
                 $input{"url"}));
    }
}
if ($input{"type"} eq "Diff" && defined($input{"version"})) {
    $extra = $input{"version"};
    if ($extra ne "penultimate" && $extra !~ /^\d+\.\d+$/) {
        &exit_gracefully("Version for diff specified as $extra not acceptable", 1);

if ($debug) {
    open(STDOUT, ">&STDOUT");
    print("<PRE>\n");

$input{"type"} =~ s/ /_/g;
exec ($backend, $input{"type"}, $input{"email"}, $input{"url"}, $extra
print "<strong>Error processing command.</strong>\n";
&PrintTrailer;
```

```
# Local Variables:
# mode: perl
# End:
```

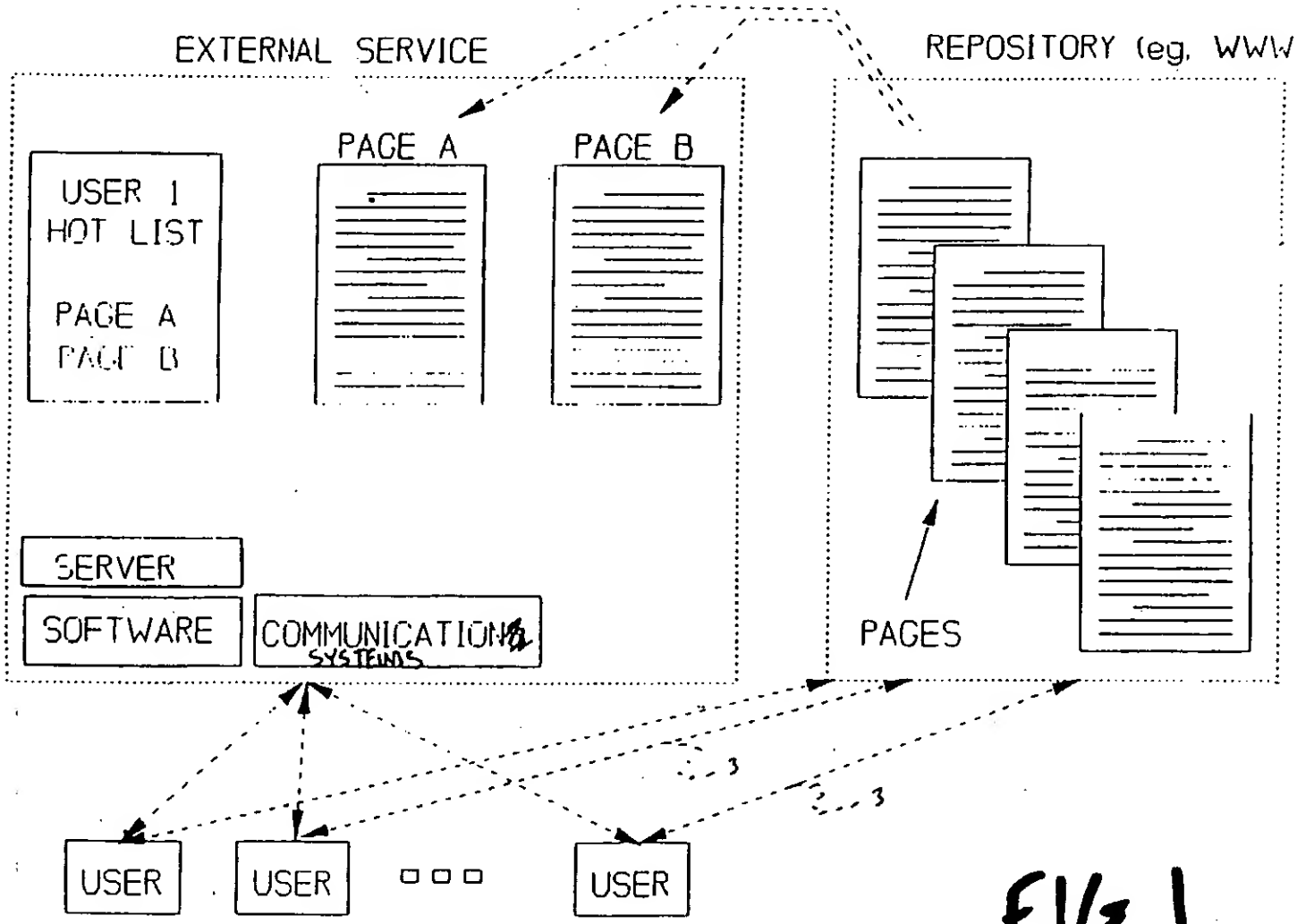

EXTERNAL SERVICE
REPOSITORY (eg. WWW)
PAGE A
PAGE B
USER 1
HOT LIST
PAGE A
PAGE B
SERVER
SOFTWARE
COMMUNICATIONS
SYSTEMS
PAGES
3
3
USER
USER
USER

FIG 1

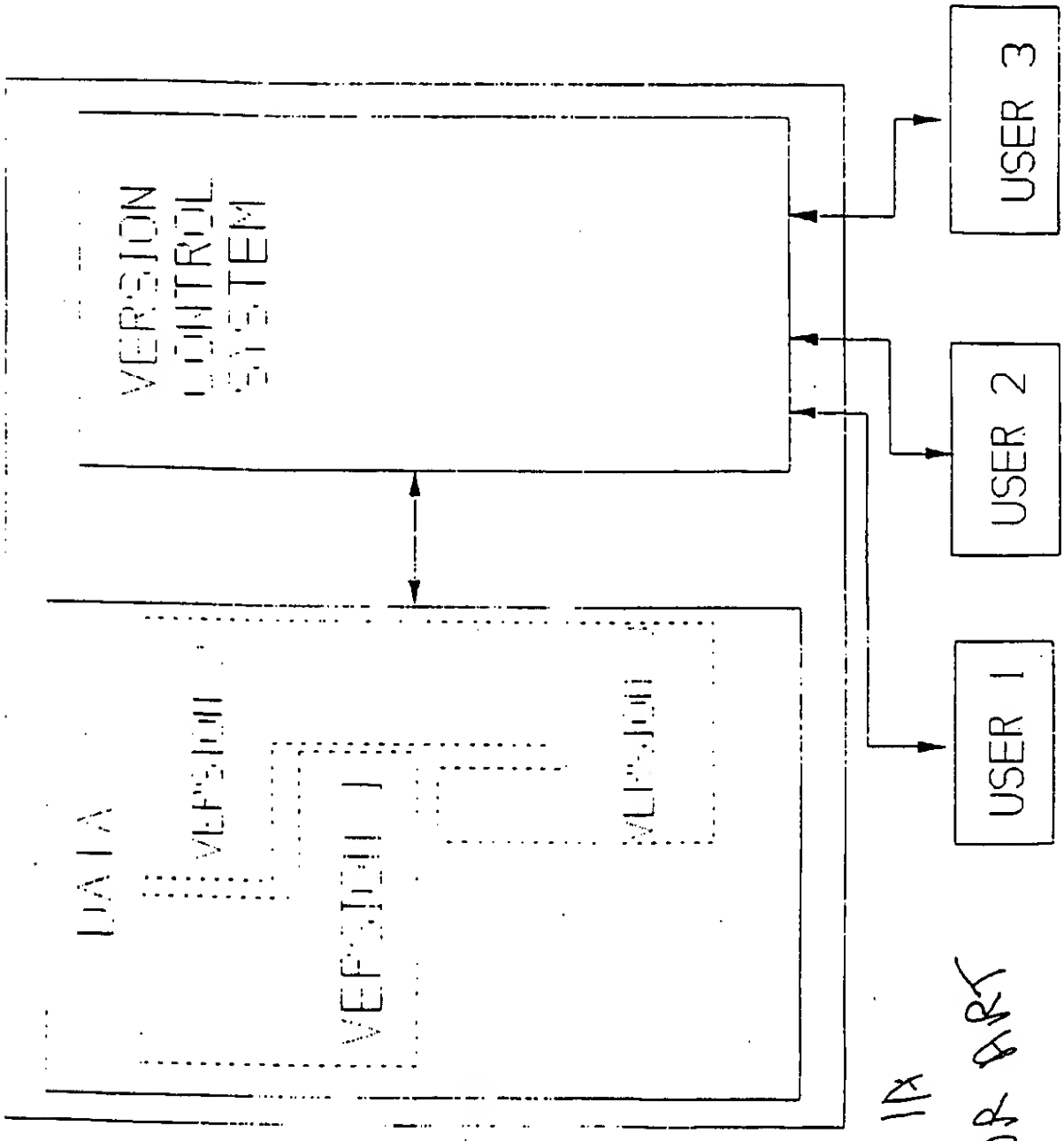


FIG 1A
PRIOR ART

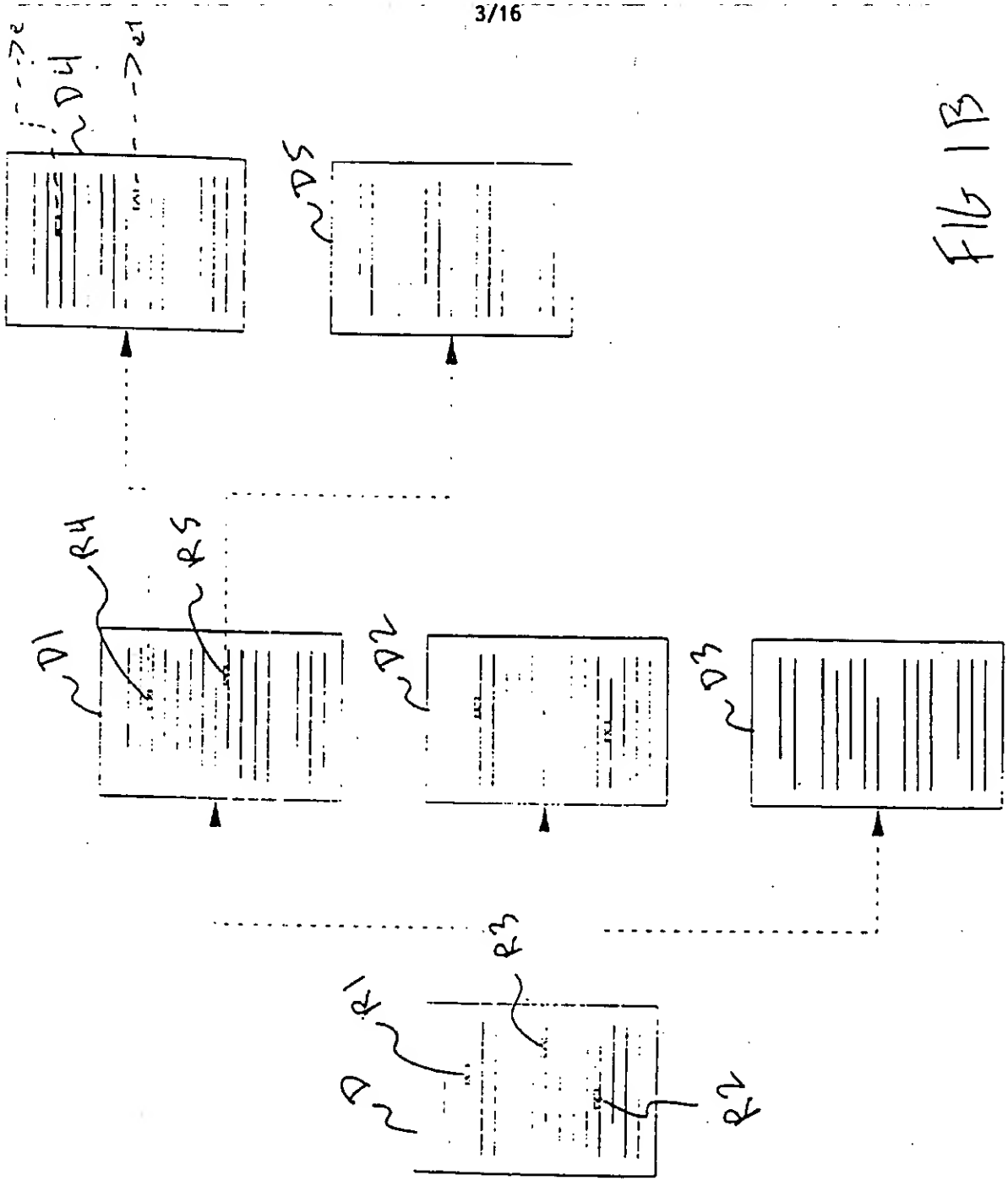


FIG 1B

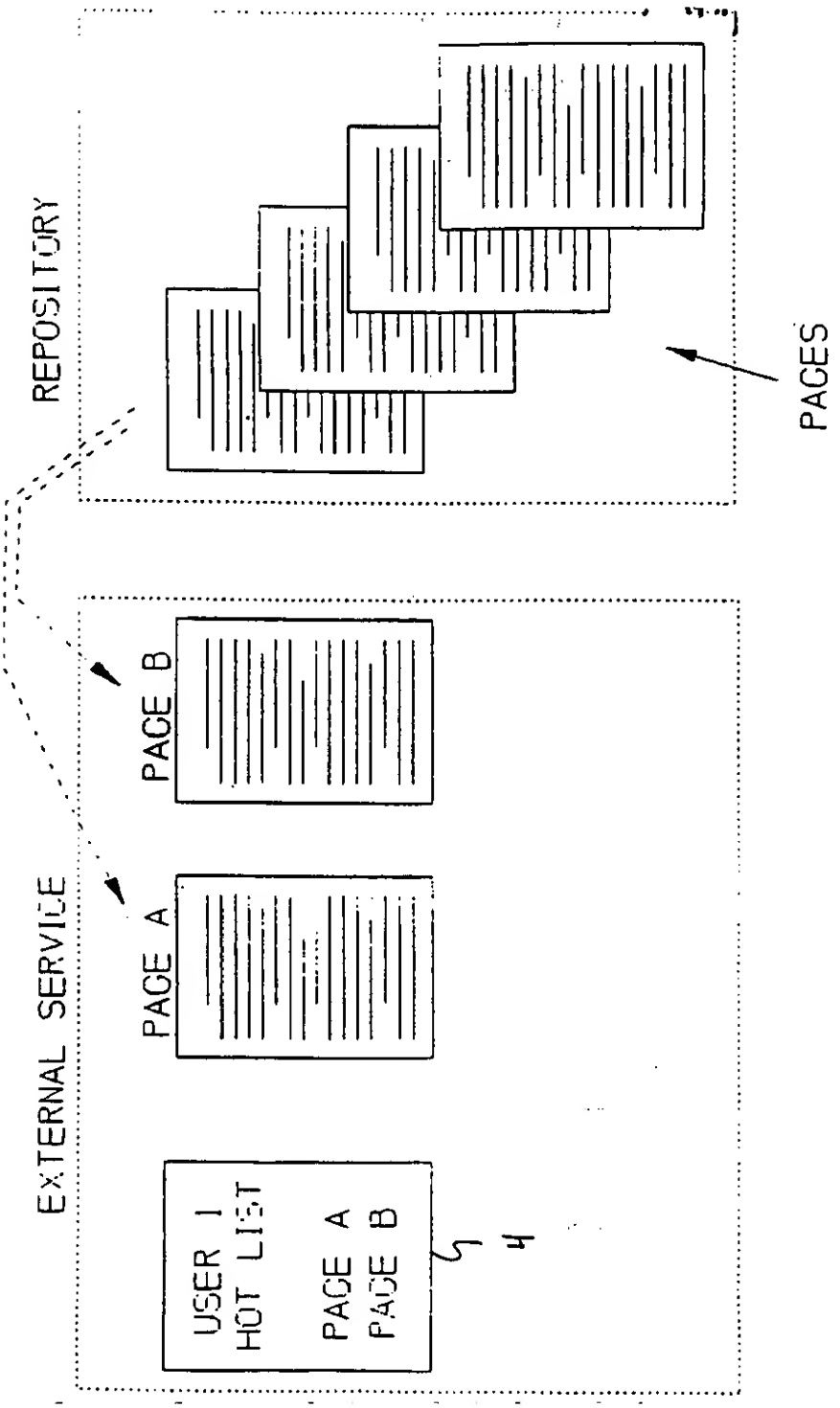

REPOSITORY
PAGES
EXTERNAL SERVICE
PAGE B
PAGE A
USER 1
HOT LIST
PAGE A
PAGE B
4

FIG 2

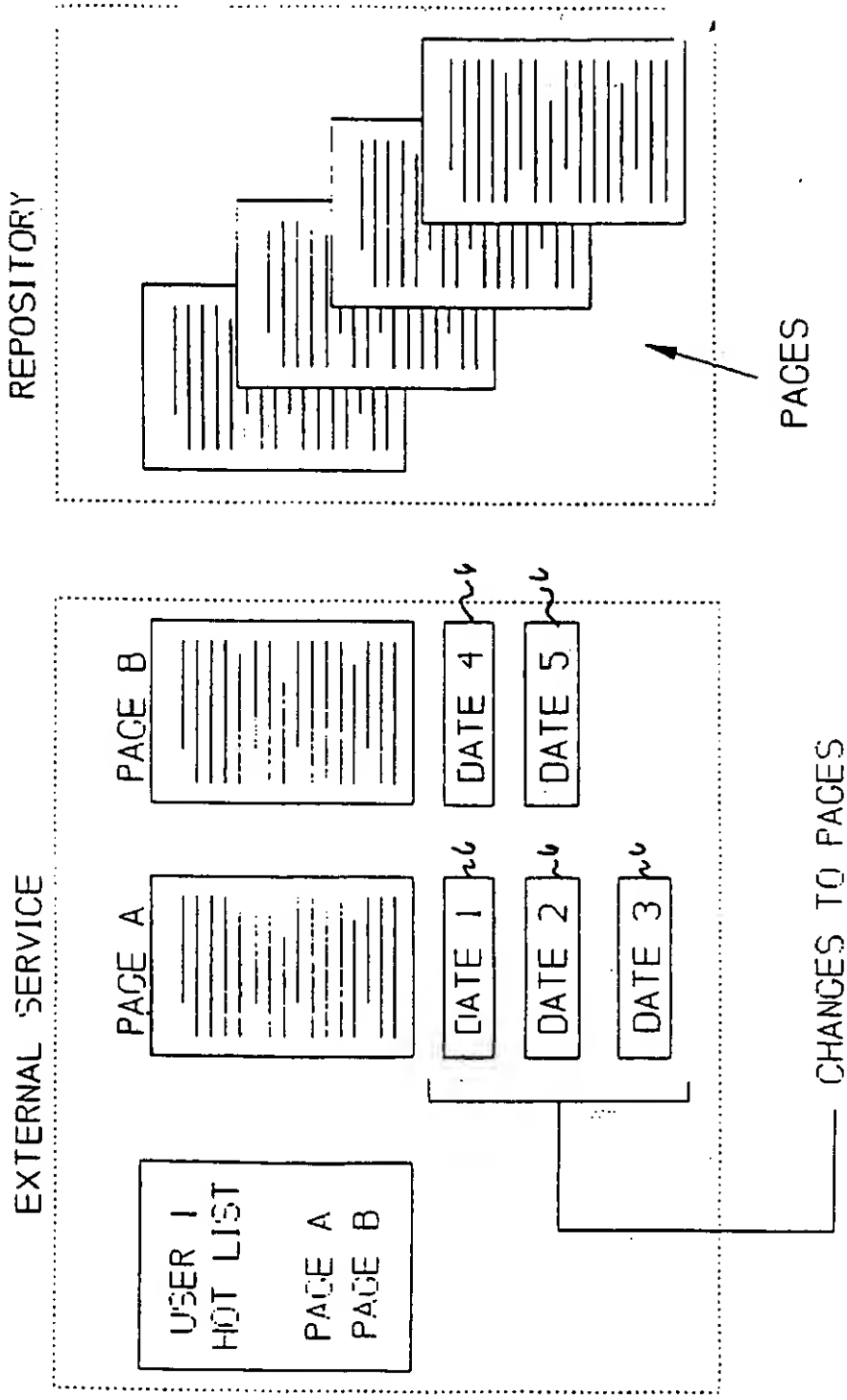

REPOSITORY
PAGES
EXTERNAL SERVICE
PAGE B
PAGE A
DATE 4
DATE 5
DATE 1
DATE 2
DATE 3
USER 1
HOT LIST
PAGE A
PAGE B
CHANGES TO PAGES

FIG 3

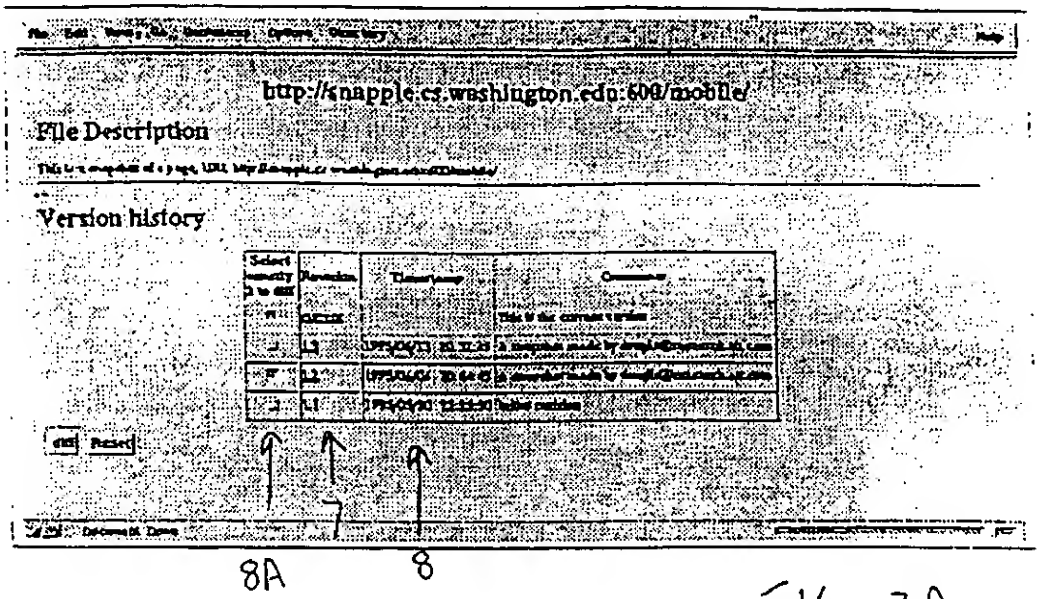
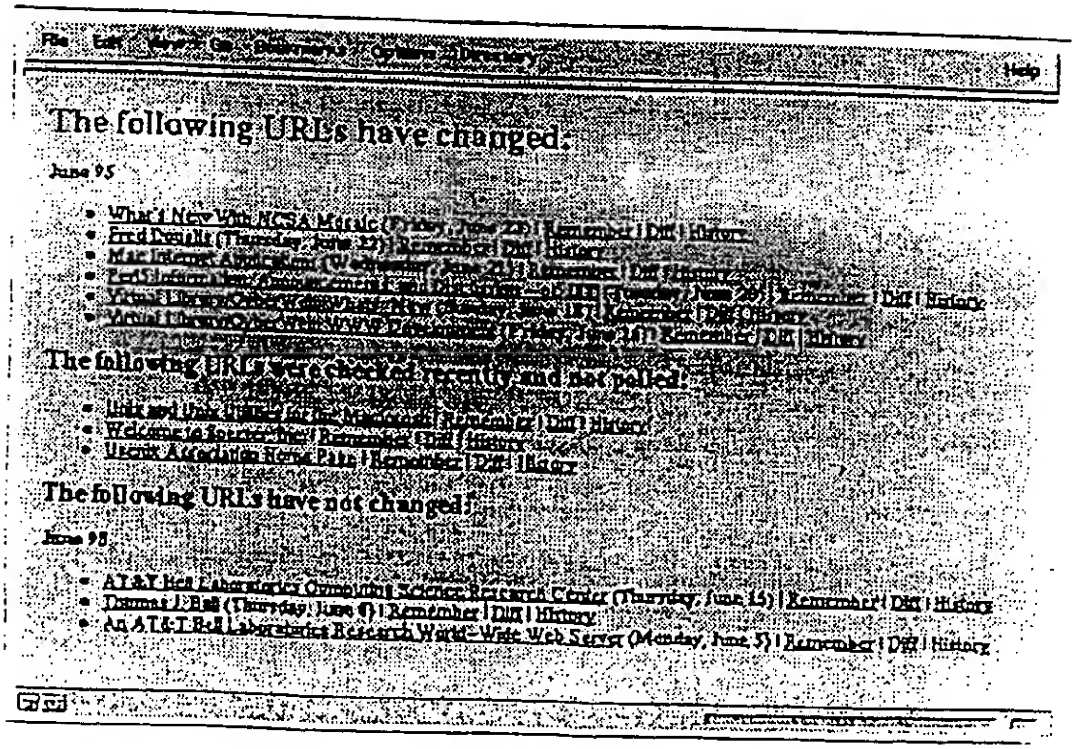

http://snapple.cs.washington.edu:600/mobile/
File Description
Version history
8A
8

FIG 3A

FIG 3B


The following URLs have changed:
June 95
The following URLs were checked recently and not polled:
The following URLs have not changed:
June 95

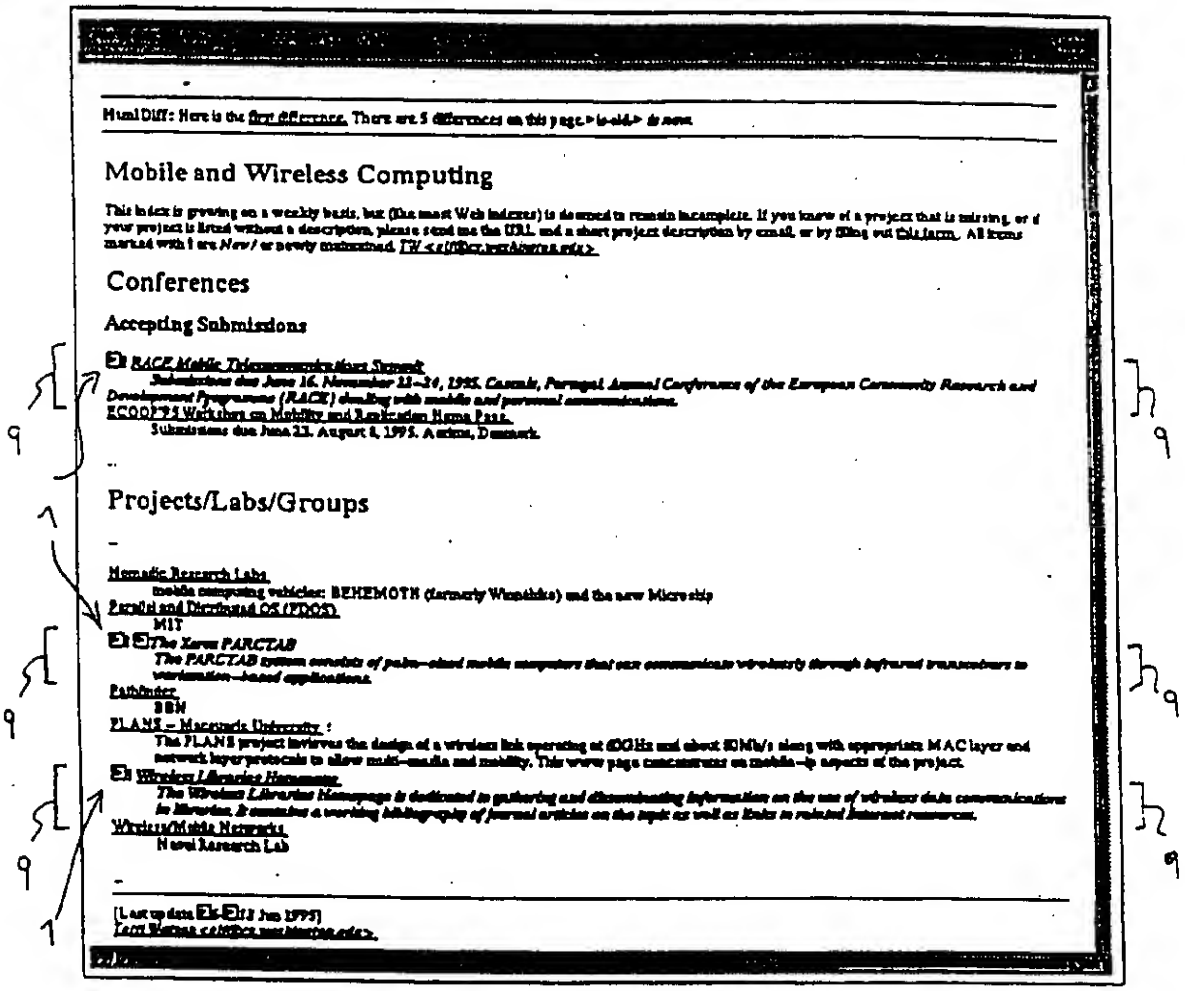

HtmlDiff: Here is the first difference. There are 5 differences on this page.
Mobile and Wireless Computing
This Index is growing on a weekly basis, but (like most Web indexes) is doomed to remain incomplete. If you know of a project that is missing, or if your project is listed without a description, please send me the URL and a short project description by email, or by filling out this form. All items marked with ! are New! or newly maintained.
Conferences
Accepting Submissions
RACE Mobile Telecommunications Summit
Projects/Labs/Groups
Nomadic Research Labs
mobile computing vehicles: BEHEMOTH (formerly Wheelbike) and the new Microship
Parallel and Distributed OS (PDOS)
MIT
The Xerox PARCTAB
Pathfinder
SRN
PLANS -- Macquarie University :
Wireless Libraries Homepages
Wireless/Mobile Networks
Nomad Research Lab
(Last update 13 Jun 1995)

FIG 4

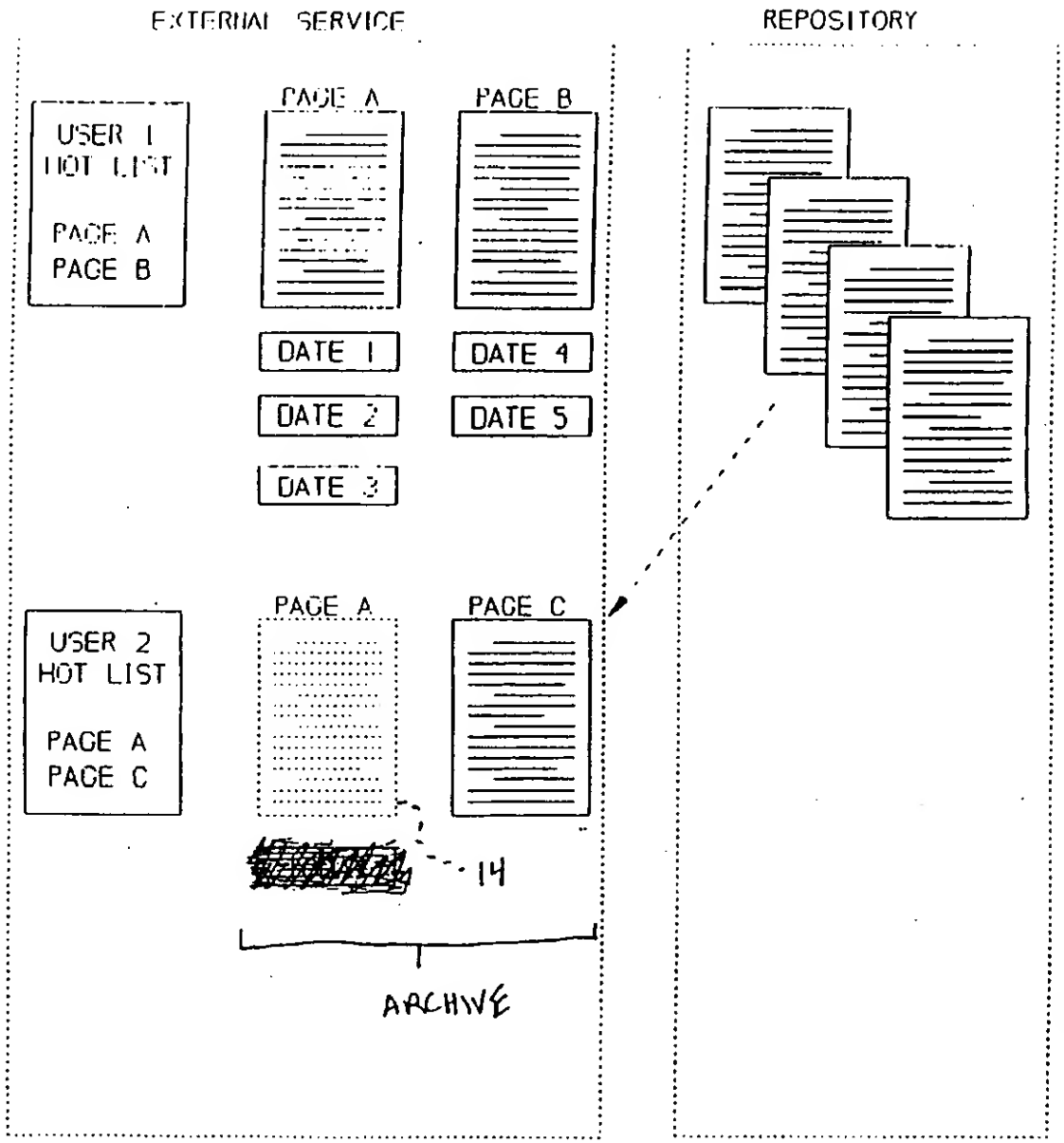

EXTERNAL SERVICE
REPOSITORY
USER 1 HOT LIST
PAGE A
PAGE B
PAGE A
PAGE B
DATE 1
DATE 2
DATE 3
DATE 4
DATE 5
USER 2 HOT LIST
PAGE A
PAGE C
PAGE A
PAGE C
14
ARCHIVE

FIG 5

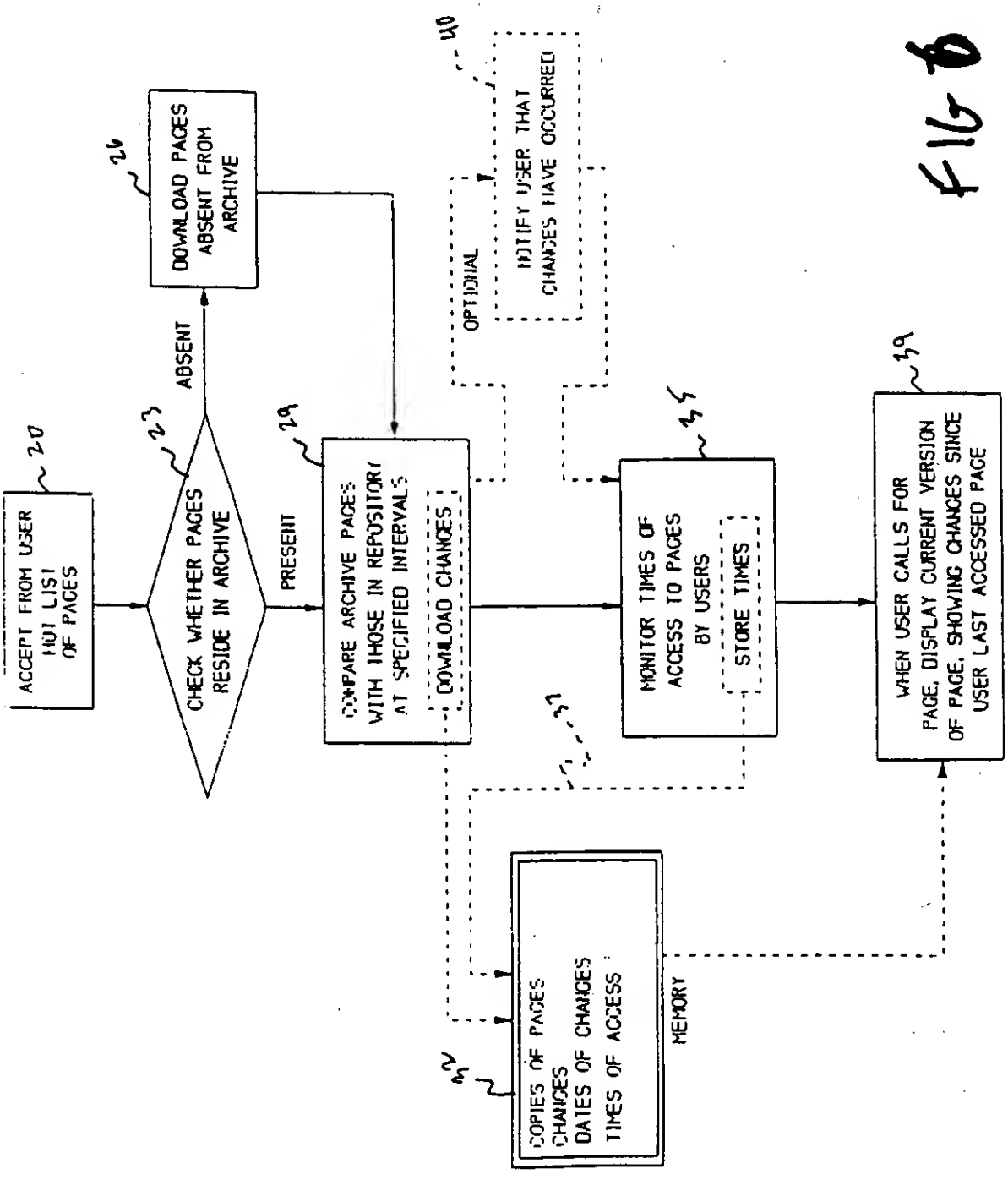

ACCEPT FROM USER HOT LIST OF PAGES
20
CHECK WHETHER PAGES RESIDE IN ARCHIVE
23
ABSENT
PRESENT
DOWNLOAD PAGES ABSENT FROM ARCHIVE
26
COMPARE ARCHIVE PAGES WITH THOSE IN REPOSITORY AT SPECIFIED INTERVALS
29
DOWNLOAD CHANGES
OPTIONAL
NOTIFY USER THAT CHANGES HAVE OCCURRED
40
MONITOR TIMES OF ACCESS TO PAGES BY USERS
45
STORE TIMES
WHEN USER CALLS FOR PAGE, DISPLAY CURRENT VERSION OF PAGE, SHOWING CHANGES SINCE USER LAST ACCESSED PAGE
49
COPIES OF PAGES
CHANGES
DATES OF CHANGES
TIMES OF ACCESS
MEMORY
37
47

FIG 6

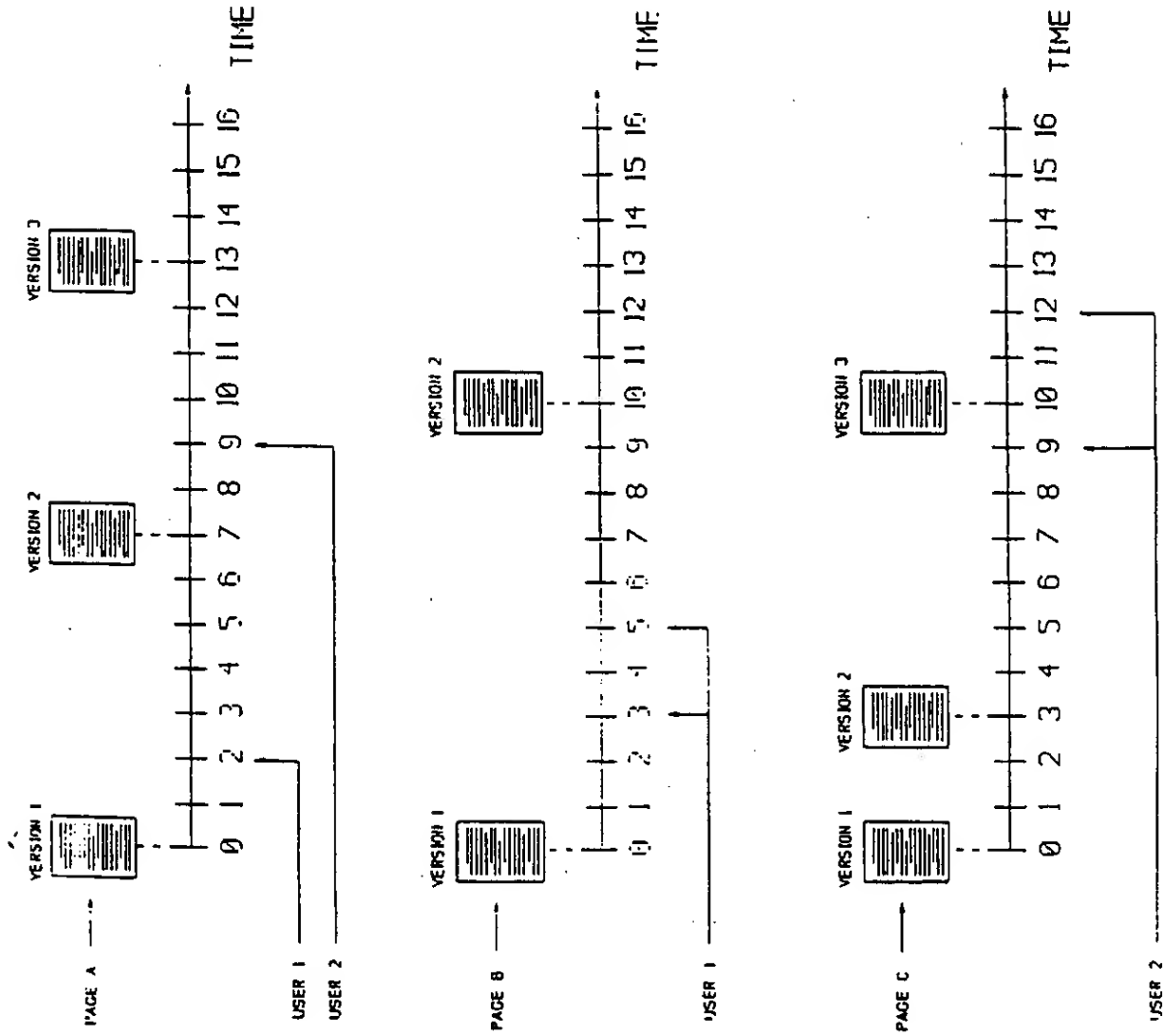

VERSION 1
VERSION 2
VERSION 3
PAGE A
USER 1
USER 2
TIME
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
VERSION 1
VERSION 2
PAGE B
USER 1
TIME
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
VERSION 1
VERSION 2
VERSION 3
PAGE C
USER 2
TIME
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

FIG 7

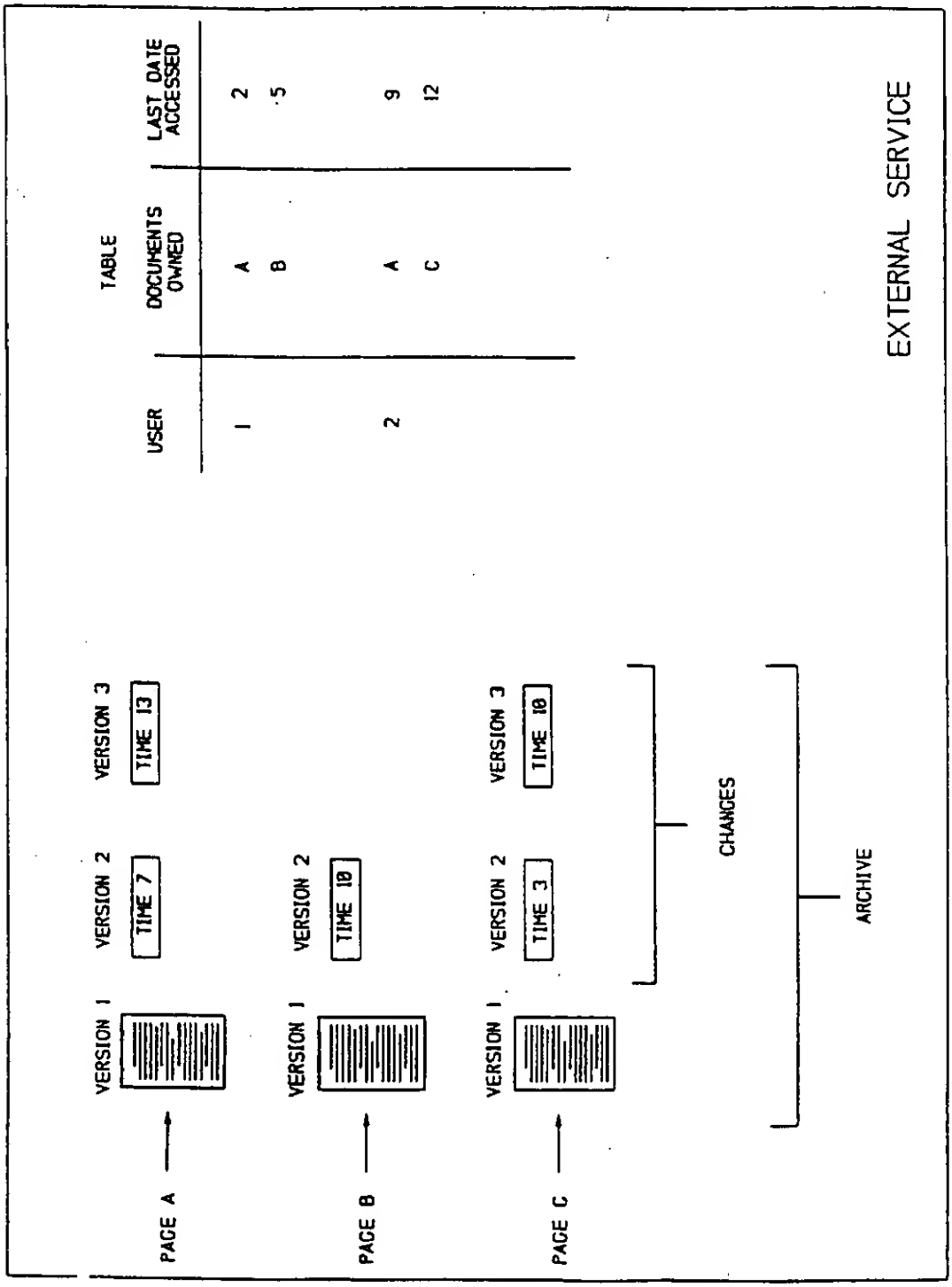

EXTERNAL SERVICE
TABLE
USER
DOCUMENTS OWNED
LAST DATE ACCESSED
1
A
2
B
5
2
A
9
C
12
PAGE A
VERSION 1
VERSION 2
TIME 7
VERSION 3
TIME 13
PAGE B
VERSION 1
VERSION 2
TIME 10
PAGE C
VERSION 1
VERSION 2
TIME 3
VERSION 3
TIME 10
CHANGES
ARCHIVE

FIG 8

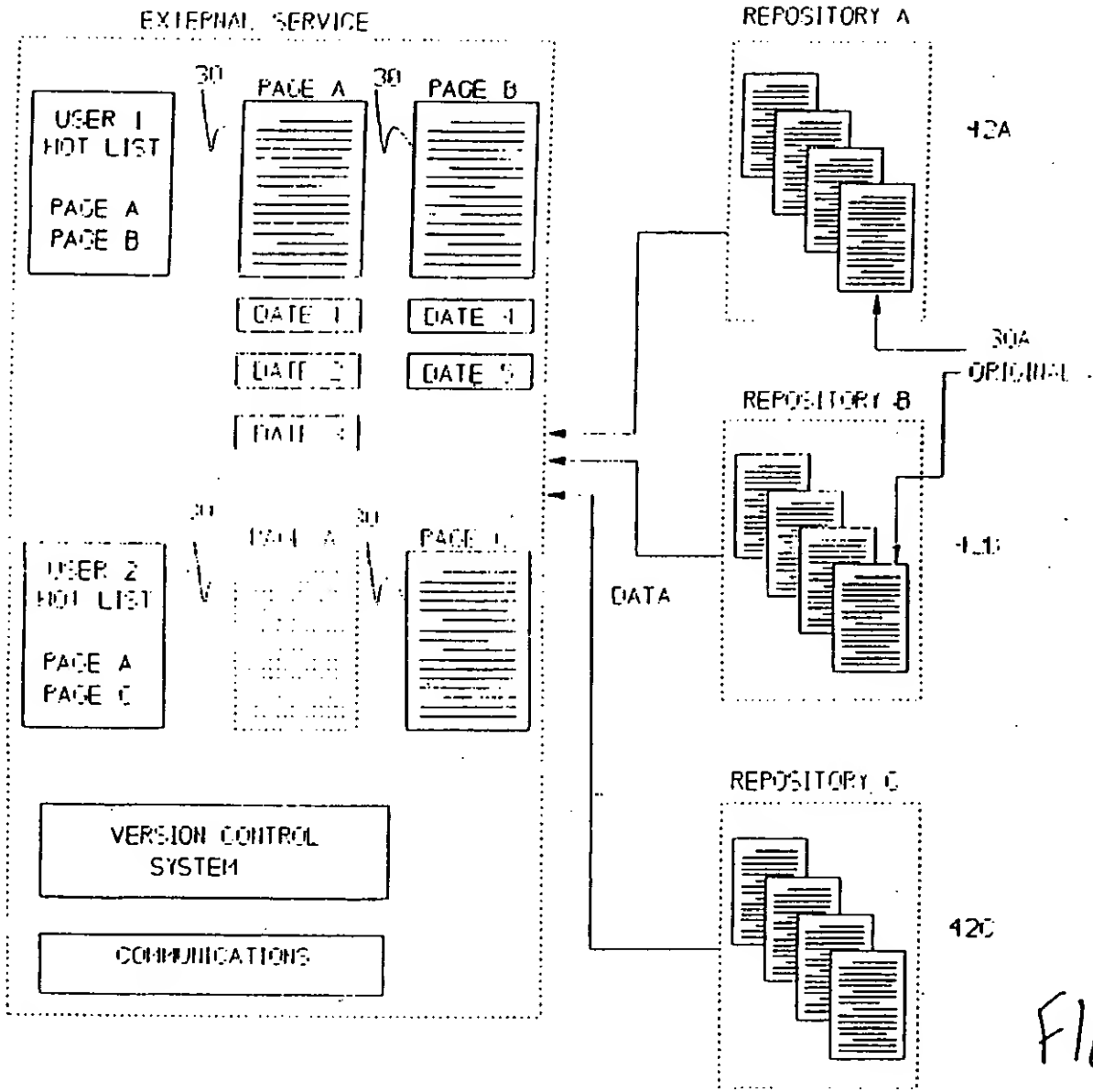

EXTERNAL SERVICE
USER 1 HOT LIST
PAGE A
PAGE B
30
PAGE A
30
PAGE B
DATE 1
DATE 2
DATE 3
DATE 4
DATE 5
USER 2 HOT LIST
PAGE A
PAGE C
30
30
VERSION CONTROL SYSTEM
COMMUNICATIONS
DATA
REPOSITORY A
42A
30A
ORIGINAL
REPOSITORY B
42B
REPOSITORY C
42C

FIG 9

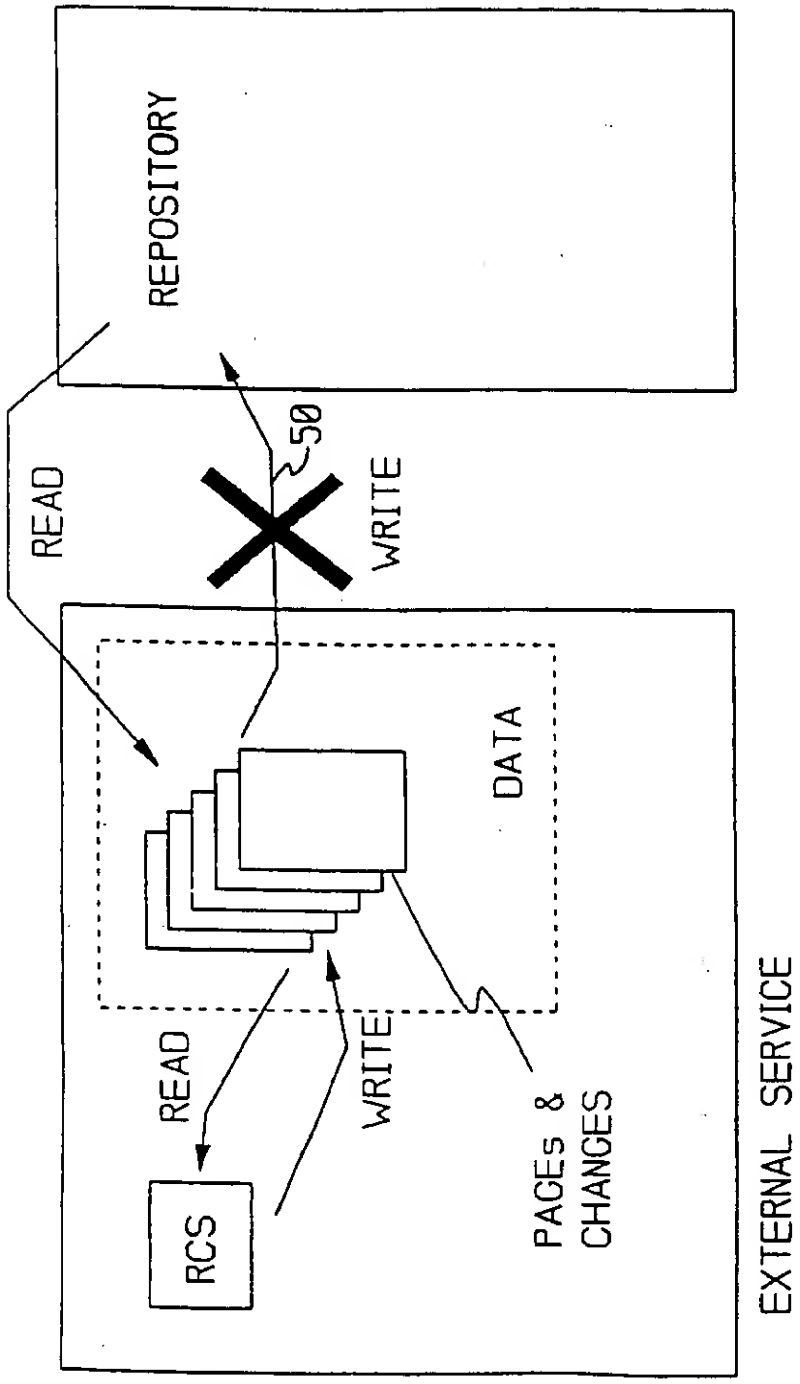
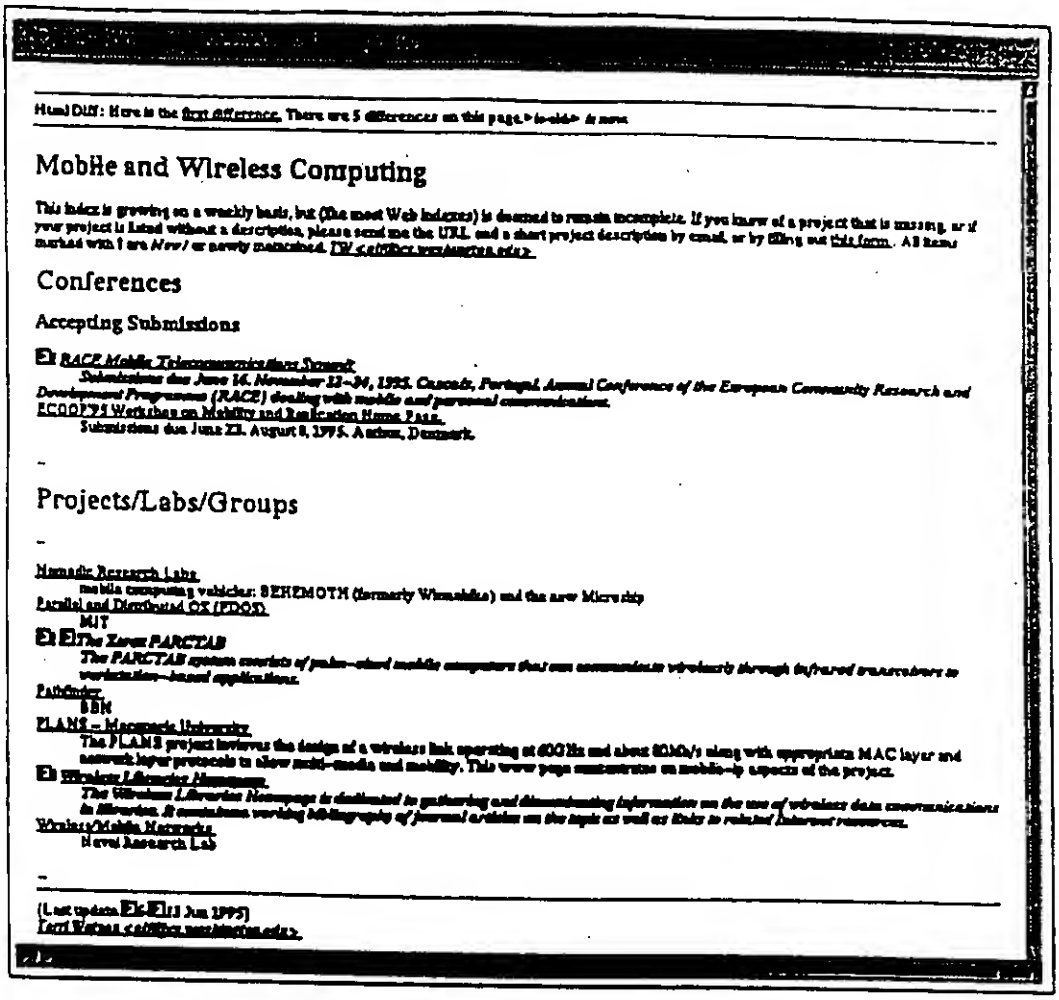

REPOSITORY
50
READ
WRITE
DATA
READ
WRITE
RCS
PAGEs &
CHANGES
EXTERNAL SERVICE

FIG 10



Mobile and Wireless Computing

This index is growing on a weekly basis, but (like most Web indexes) is doomed to remain incomplete. If you know of a project that is missing, or if your project is listed without a description, please send me the URL and a short project description by email, or by filling out this form. All items marked with ! are *New!* or newly maintained.

Conferences

Accepting Submissions

RACE Mobile Telecommunication Summit
Submissions due June 14. November 13–14, 1995. Cascais, Portugal. *Annual Conference of the European Community Research and Development Programme (RACE) dealing with mobile and personal communications.*

ECOOP'95 Workshop on Mobility and Replication Home Page.
Submissions due June 23. August 8, 1995. Aarhus, Denmark.

Projects/Labs/Groups

Nomadic Research Labs
mobile computing vehicles: BEHEMOTH (formerly Winnebiko) and the new Microchip

Parallel and Distributed OS (PDOS)
MIT

The Xerox PARCTAB
The PARCTAB system consists of palm-sized mobile computers that can communicate wirelessly through infrared transceivers to workstations—based applications.

Pathfinder
IBM

PLANS – Macromedia University
The PLANS project involves the design of a wireless link operating at 600 Hz and about 80Mb/s along with appropriate MAC layer and network layer protocols to allow multi-media and mobility. This www page summarizes an mobile-ip aspects of the project.

Wireless Libraries Homepage.
The Wireless Libraries Homepage is dedicated to gathering and disseminating information on the use of wireless data communications in libraries. It contains a working bibliography of journal articles on the topic as well as links to related Internet resources.

Wireless/Mobile Networks
Naval Research Lab

(Last update 23 June 1995)

FIG 11

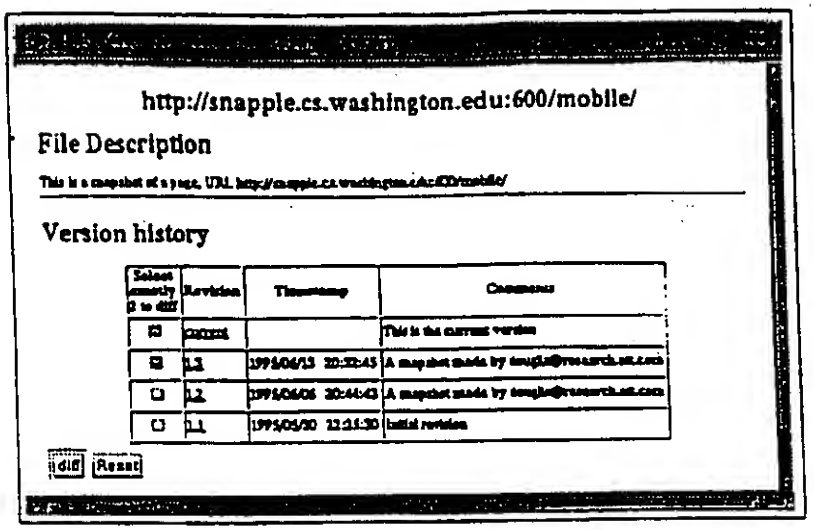
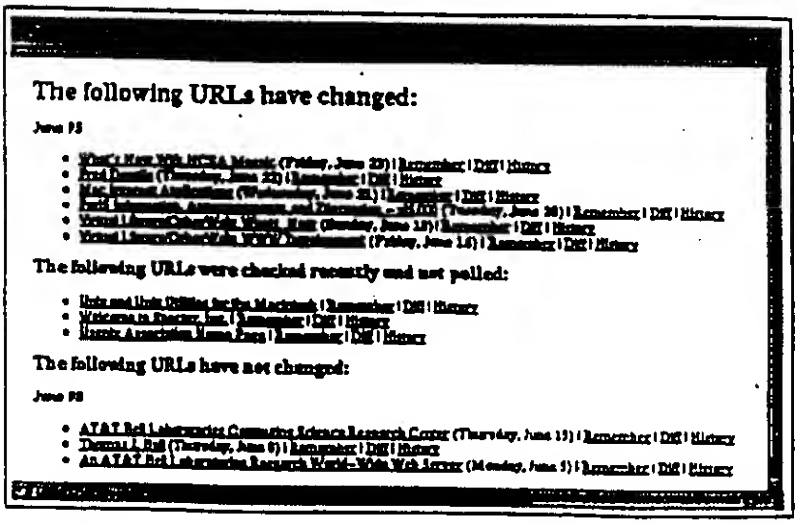

http://snapple.cs.washington.edu:600/mobile/
File Description
Version history
Select exactly 2 to diff
Revision
Timestamp
Comments
current
This is the current version
1.3
1.2
1.1
1995/05/20 12:21:20
Initial revision
diff
Reset

FIG 12

FIG 13


The following URLs have changed:
June 23
The following URLs were checked recently and not polled:
The following URLs have not changed:
June 20

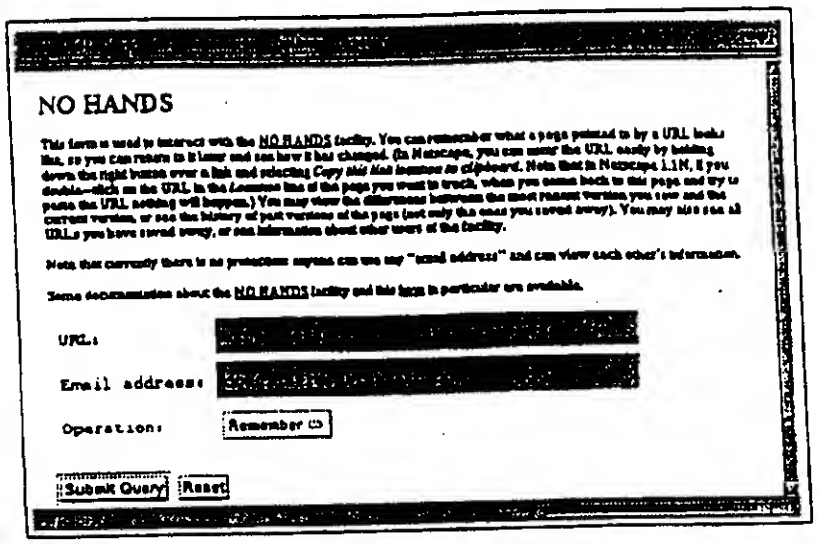

NO HANDS
This form is used to interact with the NO HANDS facility. You can remember what a page pointed to by a URL looks like, so you can return to it later and see how it has changed. (In Netscape, you can store the URL easily by holding down the right button over a link and selecting Copy this location to clipboard. Note that in Netscape 1.1N, if you double-click on the URL in the Location line of the page you want to track, when you come back to this page and try to paste the URL nothing will happen.) You may view the differences between the most recent version you saw and the current version, or see the history of past versions of the page (not only the ones you saved away). You may also see all URLs you have saved away, or see information about other users of the facility.
Note that currently there is no protection anyone can use any "email address" and can view each other's information.
Some documentation about the NO HANDS facility and this form in particular are available.
URL:
Email address:
Operation:
Remember
Submit Query
Reset

FIG 14

INTERNATIONAL SEARCH REPORT

Int ional Application No
PCT/US 96/17142

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,A	PROCEEDINGS OF THE USENIX 1996 ANNUAL TECHNICAL CONFERENCE, PROCEEDINGS OF USENIX, SAN DIEGO, CA, USA, 22-26 JAN. 1996, 1996, BERKELEY, CA, USA, USENIX ASSOC, USA, pages 165-176, XP000616939 DOUGLIS F ET AL: "Tracking and viewing changes on the Web" cited in the application see the whole document --- -/--	1-11

[X] Further documents are listed in the continuation of box C.

[ ] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
3 February 1997	14. 02 97

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl, Fax (+ 31-70) 340-3016	Katerbau, R

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inter vnal Application No
PCT/US 96/17142

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	SECOND INTERNATIONAL WORLD-WIDE WEB CONFERENCE: MOSAIC AND THE WEB, CHICAGO, IL, USA, 17-20 OCT. 1994, vol. 28, no. 1-2, ISSN 0169-7552, COMPUTER NETWORKS AND ISDN SYSTEMS, DEC. 1995, ELSEVIER, NETHERLANDS, pages 147-154, XP000616707 JONG-GYUN LIM: "Using Coollists to index HTML documents in the Web" see the whole document ---	1-11
A	PROCEEDINGS. INTERNATIONAL CONFERENCE ON TOOLS WITH ARTIFICIAL INTELLIGENCE, 1 January 1995, pages 492-495, XP000567438 PAZZANI M ET AL: "LEARNING FROM HOTLISTS AND COLDLISTS: TOWARDS A WWW INFORMATION FILTERING AND SEEKING AGENT" see the whole document ---	1-11
A	PROCEEDINGS OF THE CONFERENCE ON ARTIFICIAL INTELLIGENCE FOR APPLICATIONS, ORLANDO, MAR. 1 - 5, 1993, no. CONF. 9, 1 March 1993, INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, pages 345-352, XP000379626 BEERUD SHETH ET AL: "EVOLVING AGENTS FOR PERSONALIZED INFORMATION FILTERING" see the whole document -----	1-11

Form PCT/ISA/210 (continuation of second sheet) (July 1992)